Scientific Excellence • Resource Protection & Conservation • Benefits for Canadians
Excellence scientifique • Protection et conservation des ressources • Bénéfices aux Canadiens

The Custom Graphics Package: A Pascal Programmer's Graphics Toolbox and Data Presentation System

Brian R. Kuhn, Douglas W. A. Whitfield, and Tony Chumak

Department of Fisheries and Oceans
Biological Sciences Branch
Pacific Biological Station
Nanaimo, British Columbia V9R 5K6

December 1988

Canadian Technical Report of
Fisheries and Aquatic Sciences
No. 1659

Fisheries and Oceans
Pêches et Océans

Canada

Canadian Technical Report of Fisheries and Aquatic Sciences

Technical reports contain scientific and technical information that contributes to existing knowledge but which is not normally appropriate for primary literature. Technical reports are directed primarily toward a worldwide audience and have an international distribution. No restriction is placed on subject matter and the series reflects the broad interests and policies of the Department of Fisheries and Oceans, namely, fisheries and aquatic sciences.

Technical reports may be cited as full publications. The correct citation appears above the abstract of each report. Each report is abstracted in *Aquatic Sciences and Fisheries Abstracts* and indexed in the Department's annual index to scientific and technical publications.

Numbers 1-456 in this series were issued as Technical Reports of the Fisheries Research Board of Canada. Numbers 457-714 were issued as Department of the Environment, Fisheries and Marine Service, Research and Development Directorate Technical Reports. Numbers 715-924 were issued as Department of Fisheries and the Environment, Fisheries and Marine Service Technical Reports. The current series name was changed with report number 925.

Technical reports are produced regionally but are numbered nationally. Requests for individual reports will be filled by the issuing establishment listed on the front cover and title page. Out-of-stock reports will be supplied for a fee by commercial agents.

Rapport technique canadien des sciences halieutiques et aquatiques

Les rapports techniques contiennent des renseignements scientifiques et techniques qui constituent une contribution aux connaissances actuelles, mais qui ne sont pas normalement appropriés pour la publication dans un journal scientifique. Les rapports techniques sont destinés essentiellement à un public international et ils sont distribués à cet échelon. Il n'y a aucune restriction quant au sujet; de fait, la série reflète la vaste gamme des intérêts et des politiques du ministère des Pêches et des Océans, c'est-à-dire les sciences halieutiques et aquatiques.

Les rapports techniques peuvent être cités comme des publications complètes. Le titre exact paraît au-dessus du résumé de chaque rapport. Les rapports techniques sont résumés dans la revue *Résumés des sciences aquatiques et halieutiques*, et ils sont classés dans l'index annuel des publications scientifiques et techniques du Ministère.

Les numéros 1 à 456 de cette série ont été publiés à titre de rapports techniques de l'Office des recherches sur les pêcheries du Canada. Les numéros 457 à 714 sont parus à titre de rapports techniques de la Direction générale de la recherche et du développement, Service des pêches et de la mer, ministère de l'Environnement. Les numéros 715 à 924 ont été publiés à titre de rapports techniques du Service des pêches et de la mer, ministère des Pêches et de l'Environnement. Le nom actuel de la série a été établi lors de la parution du numéro 925.

Les rapports techniques sont produits à l'échelon régional, mais numérotés à l'échelon national. Les demandes de rapports seront satisfaites par l'établissement auteur dont le nom figure sur la couverture et la page du titre. Les rapports épuisés seront fournis contre rétribution par des agents commerciaux.

Canadian Technical Report of Fisheries
and Aquatic Sciences No. 1659

December 1988

THE CUSTOM GRAPHICS PACKAGE: A PASCAL PROGRAMMER'S GRAPHICS TOOLBOX AND DATA PRESENTATION SYSTEM

by

Brian R. Kuhn[1], Douglas W. A. Whitfield[2], and Tony Chumak[3]

Department of Fisheries and Oceans
Biological Sciences Branch
Pacific Biological Station
Nanaimo, British Columbia V9R 5K6

[1] Pacific Biological Station
[2] Whitfield Software Services, Hornby Island, B.C. V0R 1Z0
[3] Lateral Technologies, Hornby Island, B.C. V0R 1Z0

89-02361/

(c)Minister of Supply and Services Canada 1988

Cat. No. FS 97-6/1659E ISSN 0706-6457

Correct citation for this publication:

Kuhn, B.R., D.W.A. Whitfield, and T. Chumak. 1988. THE CUSTOM GRAPHICS PACKAGE: A Pascal programmer's graphics toolbox and data presentation system. Can. Tech. Rep. Fish. Aquat. Sci. 1659: xii + 150 p.

TABLE of CONTENTS

LIST of FIGURES viii

LIST of TABLES ix

LIST of PROGRAM LISTINGS x

ABSTRACT xi

FOREWORD xii

INTRODUCTION 1
- 1. Background information 1
- 2. General overview 2
- 3. How to use this manual 3
- 4. Turbo Pascal versions 4 and 5 5
- 5. Copying/Installing the package 5

PART I - CUSTOM GRAPHICS PACKAGE (CGP) 8

Chapter 1 - CGP OVERVIEW 9
- 1.1. Data types 9
- 1.2. Error messages 9
- 1.3. Missing data 10
- 1.4. Using text in graphics 10
- 1.5. Screen regions 11
- 1.6. Viewports and Clipping 11
- 1.7. World Coordinate systems 11
- 1.8. Aspect ratios 12
- 1.9. Continuous and Discrete variables 12
- 1.10. Axes, Tick marks, and Layout 12
- 1.11. Color support 13
- 1.12. Memory requirements 13
- 1.13. Screen image hard copy printing 13

Chapter 2 - KERNEL PLOTTING SERVICES 14
- 2.1. Relationship to Turbo Pascal 15
- 2.2. Hard copy printing 15
- 2.3. Specifications 15
  - 2.3.1. Dependencies 16
  - 2.3.2. Error handling services 17
  - 2.3.3. Initialization and Loading services 17
  - 2.3.4. Region, Viewport, World support services 20
  - 2.3.5. Graphics cursor movement 22
  - 2.3.6. Point oriented services 22
  - 2.3.7. Line drawing services 23
  - 2.3.8. Polygons and Shape filling services 24
  - 2.3.9. Circle, Arc, and Curve services 25
  - 2.3.10. Text writing services 26

Chapter 3 - BASIC PLOTTING SERVICES 28
3.1. Dependencies 28
3.2. Missing data 29
3.3. Label sizes 29
3.4. Plot box color control 29
3.5. Initialization services 29
3.6. Axis services 30
3.7. Data manipulation services 32
3.8. Text services 33
3.9. Symbol services 37
3.10. Miscellaneous routines 38

Chapter 4 - TWO DIMENSIONAL PLOTTING SERVICES 39
4.1. Overview 39
4.2. Specifications 42
4.2.1. Dependencies 42
4.2.2. Plot limits 42
4.2.3. PlaneBox 43
4.2.4. Continuous - Continuous plotting 44
4.2.5. Discrete - Continuous plotting 46
4.2.6. Histogram 48
4.2.7. LabelPoints 51
4.2.8. XErrorBar and YErrorBar 51
4.2.9. Contour 52

Chapter 5 - THREE DIMENSIONAL PLOTTING SERVICES 54
5.1. Overview 54
5.2. Specifications 59
5.2.1. Dependencies 59
5.2.2. CornerBox 59
5.2.3. Lolliplot 59
5.2.4. LabelLolliPoints 60
5.2.5. Surface 60

PART II - INTERACTIVE GRAPHICS SYSTEM (IGS) 61

Chapter 6 - IGS INTRODUCTION 62

Chapter 7 - IGS PROGRAMMER SERVICES 65
7.1. Overview 65
7.2. IGS Main Unit 66
7.2.1. IGSMainMenu 66
7.2.2. IGSRelease 69
7.2.3. SaveSession 69
7.2.4. LoadSession 70
7.2.5. SavePrefs 70
7.2.6. LoadPrefs 70
7.2.7. StorePointerInReal 70
7.3. An example program 71
7.4. Dependencies 73

Chapter 8 - IGS USER MANUAL 74
- 8.1. Menus and Lists 74
- 8.2. Edit Screens 74
  - 8.2.1. Text/Numeric fields 74
  - 8.2.2. Rotary fields 75
  - 8.2.3. Range fields 75
- 8.3. Main Menu 76
  - 8.3.1. Ranges 76
    - 8.3.1.1. Select 76
    - 8.3.1.2. List 77
  - 8.3.2. Preferences 78
  - 8.3.3. Exit (saving) 79
- 8.4. Graph Menu 80
- 8.5. Plot specifications 80
  - 8.5.1. Named ranges as plot variables 81
  - 8.5.2. Options screen 82
  - 8.5.3. Plot definition screens 84
    - 8.5.3.1. X-Y plot 85
    - 8.5.3.2. Discrete-Continuous plot 89
    - 8.5.3.3. Histogram plot 90
    - 8.5.3.4. Lollipop plot 92

PART III - CUSTOM PLOTTING PACKAGE (CPP) 94

Chapter 9 - CPP INTRODUCTION 95

Chapter 10 - CPP USER MANUAL 96
- 10.1. Overview 96
- 10.2. Commands 96
  - 10.2.1. New session 96
  - 10.2.2. Different Data 97
  - 10.2.3. Save session 97
  - 10.2.4. Load session 98
  - 10.2.5. Continue 98
  - 10.2.6. Exit 98
- 10.3. Preference file 98
- 10.4. Dependencies 98

Chapter 11 - CPP DATA FILE FORMAT 99
- 11.1. Data records 99
- 11.2. Field display format record 99
- 11.3. Column title record 100

PART IV - MRP GRAPHICS SYSTEM (MRPGS) 101

Chapter 12 - MRPGS INTRODUCTION 102
- 12.1. Brief overview of MRP 102
- 12.2. The input data files 102
- 12.3. Relationship to the IGS 102

Chapter 13 - MRPGS USER MANUAL 103
13.1. Overview 103
13.2. Main Menu 103
13.2.1. Layout and description 104
13.2.2. Processing the data files 105
13.3. Spreadsheet format 106
13.4. Dependencies 108

PART V - IMAGE PRINTING (PRT) 109

Chapter 14 - PRINTIMAGE DESCRIPTION 110
14.1. Introduction 110
14.2. Graphics card and mode support 111
14.3. Printer support 112
14.4. Dependencies 112

Chapter 15 - PRINTIMAGE USER MANUAL 113
15.1. Installing/Calling PrintImage 113
15.1.1. Installing the TSR version of PrintImage 113
15.1.2. Calling the TP unit version of PrintImage 114
15.2. The PrintImage Menu 116
15.2.1. Printer types 117
15.2.2. Selecting printer ports 118
15.2.3. Scan types 118
15.2.4. Printer modes 118
15.2.5. Scaling factors 119
15.3. Removing the TSR version of PrintImage from memory 119

Appendix A - TECHNICAL SPECIFICATIONS 121

Appendix B - PROCEDURE/FUNCTION REFERENCE 123

Appendix C - PLOTTING SHELLS 126
C.1. The X-Y plotting shell 126
C.1.1. The X-Y plotting shell user template 126
C.1.2. The X-Y plotting shell status and menu screen 128
C.1.3. Variables and services exported by the X-Y plotting shell 130
C.2. The Surface/Contour plotting shell 132
C.2.1. The user template for the Surface/Contour plotting shell 132
C.2.2. The Surface/Contour plotting shell status and menu screen 135
C.2.3. Variables and services exported by the Surface/Contour plotting shell 137

Appendix D - ERROR MESSAGES 139

Appendix E - DISTRIBUTION DISKS 141

References 143

Acknowledgements 144

Index 145

LIST of FIGURES

Figure 4.1 - Example of 2 y axes 40
Figure 5.1 - Example of a lollipop plot 55
Figure 5.2 - Example of a surface plot 57
Figure 6.1a - Example of the IGS opening screen 63
Figure 6.1b - Example of an IGS graph setup screen 63
Figure 6.1c - Example of an IGS Discrete - Continuous graph 64
Figure 8.1 - IGS Preferences screen 79
Figure 8.2 - Graph Menu: Options screen 82
Figure 8.3 - Range name selection screen 84
Figure 8.4a - X-Y plot definition screen example part 1. 86
Figure 8.4b - X-Y plot definition screen example part 2. 87
Figure 8.5 - Example of an IGS X-Y plot 88
Figure 8.6 - Discrete-Continuous plot definition screen 89
Figure 8.7 - Histogram plot definition screen 90
Figure 8.8 - Example of an IGS Histogram plot 91
Figure 8.9 - Lollipop plot definition screen 92
Figure 8.10 - Example of an IGS Lollipop plot 93
Figure 10.1 - Custom Plotting Package FILE menu 96
Figure 13.1 - MRP Graphics System: Main Menu 104
Figure 13.2 - MRP Graphics System example IGS spreadsheet 106
Figure C.1 - Example line graph from the X-Y plotting shell 128
Figure C.2 - Status and Menu screen for the X-Y plotting shell 129
Figure C.3 - Example surface diagram from the Surface/Contour plotting shell 134
Figure C.4 - Example contour plot from the Surface/Contour plotting shell 135
Figure C.5 - Status and Menu screen for the Surface/Contour plotting shell 136

LIST of TABLES

Table 3.1 - List of Axis plotting options for continuous variables . . . 31
Table 3.2 - List of font-use name constants 34
Table 3.3 - List of font name constants (scaled: 1.0 x 1.0) 34
Table 3.4 - List of plotting symbols 38
Table 4.1 - List of PlaneBox layout options 44
Table 4.2 - List of fill styles 46
Table 4.3 - List of Contour option codes 53
Table 5.1 - List of CornerBox layout options 59
Table 7.1 - Column data format options 69
Table 7.2 - Data file for the IGS example program 71
Table 7.3 - IGS and CGP support units required for compilation 73
Table 8.1 - Text/Numeric fields: Editing keys 75
Table 8.2 - Range selection keys 77
Table 8.3 - IGS Graphics Options: Graph Mode options 83
Table 13.1 - List of Sort field Menu / Column names 107
Table 13.2 - List of Data field Menu / Column names 107
Table 14.1 - List of Graphics cards and modes supported by PrintImage . . 111
Table 15.1 - PrintImage procedure parameter values 115
Table 15.2 - PrintImage Menu valid keystrokes 117

LIST of PROGRAM LISTINGS

Listing 4.1 - Example of 2 y axes 41
Listing 5.1 - Example of a lollipop plot 56
Listing 5.2 - Example of a surface plot 58
Listing 7.1 - A complete IGS example program listing 72
Listing C.1 - User template for the X-Y plotting shell 127
Listing C.2 - User template for the Surface/Contour plotting shell . . . 133

ABSTRACT

Kuhn, B.R., D.W.A. Whitfield, and T. Chumak. 1988. THE CUSTOM GRAPHICS PACKAGE: A Pascal programmer's graphics toolbox and data presentation system. Can. Tech. Rep. Fish. Aquat. Sci. 1659: xii + 150 p.

This report documents a custom graphics package developed for microcomputers. The package provides tools for programmers to generate sophisticated data presentation applications. Several different entry points to the package are available to provide users (with varying degrees of expertise), efficient access to the graphics functions.

A stand-alone program is provided which integrates most of the plotting services into a menu driven environment with data import capabilities. A second stand-alone program, specifically designed to interface with data files extracted from the Mark Recovery Program database (Kuhn et al. 1988 and Kuhn 1988) is also provided.

Provision for producing hard copy of graphics screen images is included with the package. This capability may be incorporated directly into user-developed applications. Alternatively, this printing function may be loaded into the computer's memory and then activated from most other programs that display graphic output.

RÉSUMÉ

Kuhn, B.R., D.W.A. Whitfield, and T. Chumak. 1988. THE CUSTOM GRAPHICS PACKAGE: A Pascal programmer's graphics toolbox and data presentation system. Can. Tech. Rep. Fish. Aquat. Sci. 1659: xii + 150 p.

Le présent rapport décrit un progiciel graphique personnalisé conçu pour les micro-ordinateurs. Ce progiciel fournit aux programmeurs les outils nécessaires pour créer des applications sophistiquées axées sur la présentation des données. Le logiciel comporte plusieurs points d'entrée différents, de sorte que des usagers dont le niveau d'expertise varie puissent accéder efficacement aux fonctions graphiques.

La plupart des fonctions de traçage sont regroupées dans un programme autonome, piloté par menu, qui offre la possibilité d'importer des données. Un autre programme autonome est conçu spécialement pour fournir une interface avec les fichiers extraits de la base de données relative au Programme de reprise de poissons étiquetés (Kuhn et al. 1988 et Kuhn 1988).

Le progiciel permet également de produire des copies en papier des images graphiques affichées sur l'écran. Les usagers peuvent incorporer cette possibilité directement dans les applications qu'ils conçoivent. La fonction d'impression peut aussi être chargée en mémoire et déclenchée à partir de la plupart des autres programmes qui affichent des sorties graphiques.

FOREWORD

B. Kuhn developed the initial specifications for, and was the scientific authority in charge of this development project. D. Whitfield of Whitfield Software Services, a co-author of this manual, submitted a bid, and won the development contract. He subsequently divided the project into task units and sub-contracted some of the programming to T. Chumak of Lateral Technologies. Together, they accomplished an exceptional amount of quality work in just two months time.

T. Chumak developed the interactive user interface code, including that for spreadsheet management and display, screens and menus. D. Whitfield wrote the graphics programs and the data manipulation code for the Mark Recovery Program Graphics System. Together, they wrote some preliminary documentation which contributed to the writing of this manual.

The product delivered by Whitfield Software Services did not include the capability for printing screen images on hard copy devices. B. Kuhn, with Dan Nesbitt, developed a flexible image printing module that programmers may use in their graphics programs. Dan was a summer student working for B. Kuhn during the months May through August, 1988.

INTRODUCTION

1. Background information

Since the advent of microcomputers, there have been many graphics packages available on the commercial market. Early on, Salmon Stock Interpretation Unit staff of the Biological Sciences Branch had custom packages developed under service contracts, but without complete success. Neither the commercial nor custom products available were very satisfactory. There are several factors that contributed to the limited quality, flexibility, and completeness of previously used packages. These include hardware inadequacies, large software modules, complicated programmer or user interfaces, and an incomplete set of functions.

Now that the microcomputer standard appears to be an 80286 type computer with high resolution (640 x 350) graphics display capability, Stock Interpretation staff decided to design and develop a custom graphics package once again. This package, which we will refer to as the CGP, is designed in a layered fashion to provide different entry points for varied user requirements. There are four layers of software as well as two stand-alone programs. The four layers are Turbo Pascal units[1] dedicated to: (1) kernel plotting services (KPS), (2) basic plotting services (BPS), (3a) two-dimensional plotting services (2DPS), (3b) three-dimensional plotting services (3DPS), and (4) integrated interactive graphics system services (IGS). Two implementations of the IGS are a custom plotting package (CPP) and a Mark Recovery Program (MRP) graphics system (MRPGS). The entire package is written completely in Turbo Pascal version 5 (Borland 1988a and 1988b) code (fully version 4 compatible). No assembly language was required to complete this project.

The KPS provides an interface between the programmer and the graphics display hardware. It uses much of the Turbo Pascal Graph Unit (GU) (Borland 1988b). The KPS provides a greater level of independence from the graphics hardware than does the GU by dimensioning the display screen using normalized coordinates (0-1) rather than physical pixel locations.

The higher level units BPS, 2DPS, and 3DPS are built on top of the KPS and the GU. A program may use services from the GU, KPS, BPS, 2DPS, 3DPS in any desired (and self-consistent) mix. However, direct calls to procedures in the GU are (usually) needed only for color control.

Programmers can develop scientific analysis software and quickly add graphic capabilities to their programs using the CGP. Some of the capabilities provided include graphs of continuous and discrete variables, with linear, logarithmic, and probability scale axes, with one or two independent Y axes, with lines, symbols, or bars. Also, lollipop plots (3D scatter plots), contour plots, and surface view plots are provided. The programmer has substantial control over layout and labelling details.

---

[1] A unit is a program module which is separately compiled. It can contain both specifications and code.

The programmer can interface his software directly to the IGS by passing an array of data. This allows sophisticated graphic capabilities to be easily incorporated into many kinds of scientific analyses without much programming overhead. Other users, who wish to create quick graphic displays for a particular data-set, can use the CPP which does not require any programming abilities. The IGS and CPP are designed with a user interface similar to that of the Lotus 1-2-3 (tm[2]) spreadsheet package; i.e., it uses a menu style similar to 1-2-3 (tm), and allows data range selections from data organized in a spreadsheet format.

Users of the MRP database (Kuhn et al. 1988) have the capability of producing graphic output for subsets of the database. The MRPGS imports a flat file extracted using the MRP-Reporter software (Kuhn 1988) on the Pacific Biological Station (PBS) VAX computer. The MRPGS utilizes the IGS for producing graphic output.

Graphic displays can be printed via an image printing facility (PRT). This capability is available as a Terminate and Stay Resident utility or via a call to a Pascal procedure. Output is possible for Epson compatible dot matrix printers as well as laser printers compatible with the Hewlett-Packard LaserJet Series II.

2. General overview

The CGP is organized in such fashion as to provide graphics services for different purposes. A programmer may use the KPS and GU exclusively to write fully customized graphics oriented programs. Furthermore, the KPS includes many of the services found in the Turbo Pascal Graph unit with one difference, they accept world coordinates in contrast to physical pixel locations; in the KPS these services have names that are prefixed by the letter N. Programmers shouldn't require many GU services other than SetColor; the KPS is designed to replace most of the other GU services. There are, however, many predefined constants exported by the GU, such as, **RED**, **BLUE**, etc. for color specification, **DETECT**, **EGA**, **VGA**, etc. for specifying graphics drivers. Many of these constants will be listed when they pertain to CGP services. Essentially, the KPS provides graphics primitives for basic graphics operations, whereas, the BPS and higher level services have more predesigned functionality and require less programming to produce satisfactory results.

If producing plots of arbitrary data-sets is a requirement, the services provided by the predesigned plotting capabilities of the 2DPS and 3DPS, is prefered. The BPS is essentially a support unit utilized by the 2DPS and 3DPS. By specifying calls to routines (in the BPS) such as **Axis**, **Transform**, **MapFont**, etc., initial plot configuration details are provided to the plotting services of the 2DPS and 3DPS. See Listing 4.1 (p. 41) and Figure 4.1 (p. 40) for an example of some services from the BPS and 2DPS. Listings 5.1 (p. 56) and 5.2 (p. 58) and Figures 5.1 (p. 55) and 5.2 (p. 57) demonstrate some services from the BPS, 2DPS, and 3DPS.

---

[2] Lotus Development Corp.

A more automated approach to plot production is available from the IGS. A programmer need only pass reference pointers of an array to the IGS and the spreadsheet-like interface of the IGS allows data ranges to be selected and plot definitions to be specified interactively. The IGS essentially integrates most of the services provided by the KPS, BPS, 2DPS, and 3DPS into an interactive plotting environment.

To utilize the plotting capabilities of the IGS without writing computer code, users may run the CPP program. This package is basically a "front-end" for the IGS with data import functions. The user must supply the CPP with the name of a data file to load, and then instruct the CPP to deliver control to the IGS. The MRPGS, also a "front-end" for the IGS, is implemented specifically for the Mark Recovery Program database (Kuhn 1988 et al. and Kuhn 1988) and also doesn't require any programming skills.

As a bonus, a graphics screen printing utility is provided with the CGP. This program (PrintImage) is implemented completely independent of the GU and the CGP. It is simply an additional tool to "round-out" the package.

Finally, two plotting "shells" are included with the package. These shells essentially provide a menu framework for plotting arbitrary two or three dimensional functions. The user must fill in a template with a few lines of Pascal computer code which define the function to plot and simply compile and run the program. Via a menu the user instructs the program to plot the function. The user's function may also reference an arbitrary data-set which may be loaded via a menu command. The shells are documented in Appendix C.

3. How to use this manual

Programmers should refer to Appendix B for an alphabetical list of all the plotting services delivered with the CGP. Additionally, at the beginning of each chapter dealing with the KPS, BPS, 2DPS, 3DPS, and IGS, is a list of each service provided in the unit, including the relevant parameter specifications.

This manual is divided into five parts. Part I deals with the KPS, BPS, 2DPS, and 3DPS units in detail. The IGS is dealt with in part II, while the CPP is described in part III. MRP database users should refer to part IV for a discussion of the MRPGS. Part V discusses the hard copy provisions provided by the PrintImage program (PRT).

PART I - CUSTOM GRAPHICS PACKAGE (CGP)

- Chapter one: An overview of the CGP data types, error messages, text usage, viewports, coordinate systems, continuous and discrete variables, axis support, color support, memory requirements, and screen image hard copy printing is provided.
- Chapter two: A description of the kernel plotting services is given with a list and description of each service.
- Chapter three: The basic plotting services are described, which export many constants, types, and variables, in addition to initialization routines, axis, data manipulation, text, symbol, legend, and other miscellaneous routines.
- Chapter four: All two-dimensional services, such as bar charts, histograms, line and point plots, and a contour plotter are documented.
- Chapter five: The three-dimensional services Lolliplot and Surface are documented.

PART II - INTERACTIVE GRAPHICS SYSTEM (IGS)

- Chapter six: An introduction to the IGS is given.
- Chapter seven: Programmer's specifications of the IGS are given, including a list and description of contants, types, variables, and each IGS support service in turn.
- Chapter eight: The IGS menus, lists, and editing screens are discussed. Text, numeric, and rotary fields are explained. Range selection and environment preference settings are also discussed. Full details are given for plot specification, as well.

PART III - CUSTOM PLOTTING PACKAGE (CPP)

- Chapter nine: An introduction to the CPP is given.
- Chapter 10: Each command in the CPP main menu is described.
- Chapter 11: The data file format recognized by the CPP is discussed. The file organization is described and important key fields, rows, and columns are discussed.

PART IV - MRP GRAPHICS SYSTEM (MRPGS)

- Chapter 12: A short overview of the MRP database (Kuhn et al. 1988) is given including a short description of the required input data files. Additionally, the relationship of the MRPGS to the IGS is explained.

INTRODUCTION

- Chapter 13: The MRPGS user interface layout is explained. Rotary fields are explained, as well as other aspects of running the MRPGS program.

PART V - IMAGE PRINTING (PRT)

- Chapter 14: A description of the technical aspects of image printing, including a discussion of graphics card support, printer support, and memory resident programs is provided.
- Chapter 15: Installation and activation of the PrintImage program, selection of printer and graphics card types, and other options are discussed.

4. Turbo Pascal versions 4 and 5

The CGP is built on top of the Graph Unit (GU) that is incorporated into the Turbo Pascal version 5 (and version 4) programming language. The programmer may use any of the GU functions and procedures in addition to the functions and procedures available from the CGP. This manual documents some of the GU routines and in most cases references the relevant page number in the Turbo Pascal User's (Borland 1988a) or Reference Guide (Borland 1988b). Where the text refers to "Turbo Pascal", it assumes version 4 or 5.

It is assumed that the programmer has working knowledge of Turbo Pascal version 4 or 5 (TP) and its working environment. The CGP uses the UNIT concept found in TP to implement the different entry points; i.e., the KPS, BPS, 2DPS, 3DPS, and IGS are each a separately compiled TP UNIT. The programmer need only reference the units depended on by his program (each of the CGP units have specific TP and CGP unit dependencies).

To utilize the CPP or MRPGS, users need not even be aware that these two programs are written in TP. On the other hand, programmers who want to incorporate some graphic capabilities from other CGP components must have the TP compiler environment and manuals available.

5. Copying/Installing the package

This package is available on one 1.2 Megabyte (MB) floppy diskette or four 360 Kilobyte (K) floppy diskettes. Users may wish to make copies of the diskette(s) (they are not copy protected) before proceeding with the installation procedure.

Before installing the package, a subdirectory should be created on the hard disk (possibly "\CGP") and roughly 2½ MB of space must be available on the hard disk[3]. Furthermore, the file 'Read.ME' found on the first disk, should be examined for special information pertaining to copying files, package installation, program changes, problem resolutions, and any other important items.

See Appendix E for a detailed file list to determine which files to extract. To extract the entire package, proceed with the instructions given below for each of the six executable files located on the diskette(s).

<u>The seven self extracting program files</u>:

APL!.EXE, CGP!.EXE, Demo2D!.EXE, Demo3D!.EXE MRP!.EXE, PLT!.EXE, PRT!.EXE

Type: [assuming installation to C:\CGP]

```
A: <Enter>
APL! C:\CGP
CGP! C:\CGP
Demo2D! C:\CGP
Demo3D! C:\CGP
MRP! C:\CGP
PLT! C:\CGP
PRT! C:\CGP
```

For the last disk only, type (if Turbo Pascal is not available):

```
Copy A:*.BGI C:\CGP
Copy A:*.CHR C:\CGP
```

Entering the commands listed above will extract and copy, to the CGP directory located on hard disk C, the entire package, i.e., source code, executable programs, graphics drivers, and stroked-font files.

---

[3] To store the entire package requires approximately 2½ MB, i.e. the maximum disk space needed. It is unlikely that users will require all files in the package, so the actual space requirement is probably much lower.

<u>Other possibilities</u>:

All Pascal source code files:	680 K
CGP Pascal source code files:	148 K
Demo executable program files:	514 K
All pre-compiled unit files:	508 K
All MRP files:	269 K
All PLT files:	233 K

To extract only the source code for the CGP units type

```
CGP! C:\CGP CGP*.PAS
```

or to extract the demonstration executable programs type

```
Demo2D! C:\CGP *.EXE
Demo3D! C:\CGP *.EXE
```

Appendix E provides a complete file list and description to assist the user with deciding which CGP files to extract from which diskettes.

PART ONE

CUSTOM GRAPHICS PACKAGE (CGP)

Chapter 1 - CGP OVERVIEW
Chapter 2 - KERNEL PLOTTING SERVICES
Chapter 3 - BASIC PLOTTING SERVICES
Chapter 4 - TWO DIMENSIONAL PLOTTING SERVICES
Chapter 5 - THREE DIMENSIONAL PLOTTING SERVICES

Chapter 1 - CGP OVERVIEW

This chapter provides a brief discussion of many of the graphics concepts implemented in the CGP. There are several important features programmers should be aware of, such as how REAL (floating point) numbers are treated with and without the Intel 80287 numeric processor (MATH) chip and how to encode missing data points.

Initialization is hierarchical; if only the GU is used, the TP routine InitGraph (Borland 1988b, p. 313) should be called. On the other hand if the KPS is used but not other CGP units, the KPS routine InitKPS should be called. If higher level CGP units are used, only the BPS routine InitCGP should be called.

Sections 1.2 through 1.11 list constants, variables, functions, and procedures that are relevant to the section. The TP unit where these reside is given is brackets (e.g., (KPS)) for a cross reference to the appropriate CGP unit.

1.1. Data types

Throughout the CGP, where floating point numbers are used, the routines declare these to be of type REAL. Without the MATH (80287 coprocessor) chip, REAL requires six bytes of memory storage. For use on a system with a MATH chip, REAL may be redeclared as type SINGLE, which requires four bytes of memory storage. The programmer must decide which of these to use and be consistent throughout. To create a program that can run with or without MATH chip support, this code segment must be imbedded immediately after the TP USES statement in all modules (programs and units):

```
(*$IfOpt N+*)
Type REAL = SINGLE;
(*$EndIf*)
```

The O/C/Numeric compiler switch must be set appropriately at compile time. If Turbo Pascal version 5.0 is used, the O/C/Emulate compiler switch may be activated to run programs on machines without a MATH chip while using the MATH chip data types. This causes Turbo Pascal to encorporate a MATH chip emulation library with the user's program.

1.2. Error messages

Two different kinds of error conditions can occur while using the CGP. A nonfatal error is an exception the CGP or GU can anticipate, record, and report to the programmer (or user). The second kind of error is a fatal condition, usually caused by programmer error, hardware failure/exception, or an illegal programming condition such as referencing an invalid memory location.

If a nonfatal error occurs, the CGP is prepared to "trap" it, saving it for display at the programmer's request. If a fatal error occurs (i.e., the program crashes), TP displays an error message. If the program is in a graphics mode when it crashes, the error message will not necessarily be readable. This situation (a crash) is one that the CGP does not handle for the programmer. Skilled TP programmers could write their own fatal error trapping routine that simply restores the display mode to text and passes control on to the TP error handler. The TP Owner's Handbook provides the necessary information in the section 'Exit Procedures' (Borland 1988b, p. 229) for writing a custom error handling routine.

CGP support: variable: NErrors and services: RecordError, ResetError, and PrintError. (KPS)

1.3. Missing data

Most "real world" applications have data-sets that are incomplete; i.e., they contain unknown or missing data values. The CGP provides a facility for handling these. The constants **MissingTestValue** and **MissingDataValue** are defined to support missing data. The constant **MissingDataValue** is a real number with a value larger than **MissingTestValue** and is as large a real number as can be defined. To indicate missing data in a data-set, set the missing entries equal to **MissingDataValue**. Data values greater than **MissingTestValue** are not plotted by services exported by the 2DPS and 3DPS.

CGP support: constants: MissingTestValue and MissingDataValue. (BPS)

1.4. Using text in graphics

The GU supports two methods of displaying text in a graphics mode. One method uses a bit-mapped font (character is defined in dots). The second method uses a stroked-font (character is defined by vector movements). The stroked font method is superior to the bit-mapped font for size magnifications, due to the distortions that occur when magnifying bit-mapped fonts.

The GU provides some facilities for writing text in graphics but they are awkward to use, ambiguous, imprecise, and the placement of vertically written text suffers from several bugs. The CGP resolves most of these inadequacies and compensates for the bugs. The CGP defines specific font uses (i.e. axis labelling, major titles) and provides a routine for mapping a font use onto an actual GU font at a given magnification.

CGP support: services: FontPath, MapFont, OutAText, OutATextXY, SetTextAlignment, SetFontAndDirection, CharHeight, CharDepth, StringWidth. (BPS)

1.5. Screen regions

The CGP supports one, two, or four independent equal sized drawing regions. This feature allows up to four different graphs to be displayed simultaneously. Initially, only one screen region is created, this region being the entire physical screen. All screen addressing is relative to the active screen region (i.e., each region has its own coordinate system).

CGP support: services: DefineRegion and NextRegion. (KPS)

1.6. Viewports and Clipping

A viewport is a "window" into a screen region where all drawing action will occur. It is specified in Normalized Device Coordinates (NDC), which are real numbers in the range 0.0 to 1.0, and are relative to the screen region in use. NDC's increase from left to right and bottom to top. multiple viewports may be established and referenced by numbers (0,1,..,n), with only one selected at a time. Initially, viewport number 0 is created with bounds (0.0,0.0) and (1.0,1.0), i.e., the entire physical screen. The selected viewport may be cleared or may inherit a border via parameterless procedure references.

Clipping is the process of restricting drawing actions to a viewport. If clipping is off, then the viewport bounds are not respected by subsequent drawing actions.

CGP support: services: DefineViewport, SelectViewport,
DrawVPBox, NClearViewPort, TurnClipOn, TurnClipOff. (KPS)

1.7. World Coordinate systems

By finding the minimum and maximum value for a data-set, a world coordinate system may be specified to match the data-set. At initialization, world coordinate system number 0 is established with bounds (0.0,0.0) to (1.0,1.0), and is selected. All screen locations (excluding viewport definitions) are specified in world coordinates. When a viewport is selected, the current world coordinate system is mapped on to it.

Higher level services (BPS,2DPS,3DPS) make extensive use of viewports, clipping, and world coordinate systems, and in most graphing operations, graph lines and symbols are clipped to the drawing 'box'.

CGP support: services: DefineWorld and SelectWorld. (KPS)

1.8. Aspect ratios

The aspect ratio affects drawing of symbols, stroked-font text, circles, arcs, and pie slices. Aspect ratio is the width to height ratio of a pixel (maximum Y resolution divided by maximum X resolution). Symbols and stroked-font text are compensated for the aspect ratio changes that accompany the switch, for example, from high resolution to low resolution CGA modes. No assistance is given for circle, arc, and pie slice production.

1.9. Continuous and Discrete variables

The CGP makes a clear distinction between variables which are continuous (can take on any value in some range) and those which are discrete. The latter may be non-numeric, and are called categories. The independent variable for a plot may be discrete or continuous; dependent variables are always continuous.

1.10. Axes, Tick marks, and Layout

For all plot types (from the BPS, 2DPS, and 3DPS), the programmer must specify the attributes of each axis to be used and certain overall layout details. The order these attributes are specified is unimportant. Specifying these details does not cause any screen activity to occur. When a call to one of the routines that displays lines, bars, or symbols, occurs, then axes, tick marks, borders, and the various labels are displayed.

Defining the characteristics of an axis for a continuous variable requires the minimum and maximum values, what data transformation, if any, is to be applied, and various customization options. For a discrete variable, the number of categories and their names must be provided. Additionally, for either variable type, a character string containing the label for the axis, may be specified.

With two-dimensional plots in which both the independent and dependent variables are continuous, either one or two dependent variable axes may be specified for a graph. Drawing on a graph with two axes may refer to either axis.

Considerable control over placement of axis tick marks is provided. To keep the annotation strings relatively short, the axis routines may use powers of 10 and additive offsets. The layout process leaves extra space for printing these extra annotation details.

CGP support: services: Axis and AxisD. (BPS). PlaneBox. (2DPS)

1.11. Color support

Color control is provided by the GU, therefore the CGP does not include much additional support; however, the BPS exports four variables which may be set by the programmer to influence drawing color. These control four aspects of color in a graph: field color, box color, label color, and viewport box (border) color. For more information on changing drawing colors, refer to the GU routine **SetColor** (Borland 1988b, p. 376).

> **CGP support:** variables: FieldColor, BoxColor, LabelColor, and VPBoxColor. (BPS)

1.12. Memory requirements

Much of the CGP internal variable storage is allocated dynamically. Only the procedures **PlotBarD** and **PlotBarS** use static arrays. Font loading is limited to those fonts actually used. Considering the size of the TP units in this package, programmers should be aware of the suggestions in the TP User's Guide (Borland 1988a, pp. 141-142), which describes linking of large programs. To be used effectively, a minimum configuration of an IBM PC compatible microcomputer with a fixed disk drive and 640 K of internal RAM memory should be used.

1.13. Screen image hard copy printing

Graphic displays may be printed on dot matrix and laser printers. Provision for image "stretching" and rotation is included. Color information is reduced to a black and white printout. For further details, see section 2.2 and Part V of this document.

Chapter 2 - KERNEL PLOTTING SERVICES

The CGP is designed and implemented in a layered fashion. The Kernel Plotting Services (KPS) is the bottom layer module implement on top of the TP Graph unit. It is designed to handle many of the elementary graphics tasks. All other units in the CGP are built on top of the KPS and GU. Additionally, the KPS provides for the future addition of drivers for external hard copy devices. This is accomplished by making optional calls to procedures in another unit, called the Hard Copy Services (HCS). The HCS only contains dummy routines at present. To use the KPS unit in a program, include a reference to the CGPKPS in the program's USES statement together with any other required unit references.

Service names and parameter lists	Page no.
RecordError(ErrStr : STRING);	17
ResetError;	17
PrintError;	17
InitKPS(VAR GraphDriver,GraphMode : INTEGER; DriverPath : STRING);	17
LoadDriver(FileString : STRING) : BOOLEAN;	19
LoadFont(FileString : STRING) : BOOLEAN;	19
HardCopyOn;	19
HardCopyOff;	19
DefineRegion(Nx,Ny : WORD);	20
NextRegion;	20
DefineViewport(I : INTEGER; Rx1,Ry1,Rx2,Ry2 : REAL);	20
SelectViewport(VAR I : INTEGER) : BOOLEAN;	21
DrawVPBox;	21
NClearViewPort;	21
DefineWorld(I : INTEGER; Rx1,Ry1,Rx2,Ry2 : REAL);	21
SelectWorld(I : INTEGER) : BOOLEAN;	21
TurnClipOn;	22
TurnClipOff;	22
NGetX : REAL;	22
NGetY : REAL;	22
NMoveTo(Rx,Ry : REAL);	22
NMoveRel(Rdx,Rdy : REAL);	22
NPutPixel(Rx,Ry : REAL; Pixel : WORD);	22
NGetPixel(Rx,Ry : REAL) : WORD;	22
XSizeOfPixel : REAL;	23
YSizeOfPixel : REAL;	23
NLine(Rx1,Ry1,Rx2,Ry2 : REAL);	23
NLineTo(Rx,Ry : REAL);	23
NLineRel(Rdx,Rdy : REAL);	23
NSetLineStyle(LineStyle,Pattern,Thickness : WORD);	24
NRectangle(Rx1,Ry1,Rx2,Ry2 : REAL);	24
NBar(Rx1,Ry1,Rx2,Ry2 : REAL);	24
NBar3D(Rx1,Ry1,Rx2,Ry2,Depth : REAL; Top : BOOLEAN);	24
NDrawPoly(NumPoints : WORD; VAR RPolyPoints);	25
NFillPoly(NumPoints : WORD; VAR RPolyPoints);	25
NFloodFill(Rx,Ry : REAL; Border : WORD);	25
NArc(Rx,Ry,Radius : REAL; StAngle,EndAngle : WORD);	25

Service names and parameter lists continued...	Page no.
NGetArcCoords(VAR AC : NArcCoordsType);	25
NCircle(Rx,Ry,Radius : REAL);	26
NEllipse(Rx,Ry,Xr,Yr : REAL; StAngle,EndAngle : WORD);	26
NPieSlice(Rx,Ry,Radius : REAL; StAngle,EndAngle : WORD);	26
NOutText(TextString : STRING);	26
NOutTextXY(Rx,Ry : REAL; TextString : STRING);	26
NSetUserCharSize(FactorX,FactorY : REAL);	26
NTextHeight(TextString : STRING) : REAL;	27
NTextWidth(TextString : STRING) : REAL;	27

2.1. Relationship to Turbo Pascal

The entire KPS source code is written in Turbo Pascal version 4.0 and is fully compatible with Turbo Pascal version 5.0. Some of the KPS routines provide completely new operations, while others are direct analogues of GU routines. The latter, which move the current graphics cursor, generate marks on the screen, or make enquiries about screen locations, accept arguments in world coordinates. They are distinguished from the corresponding GU routines, which accept arguments in pixels, by the prefix letter N in their names.

2.2. Hard copy printing

In addition to the future HCS support to provide true hard copy image reproduction, the program PrintImage is supplied. PrintImage can function in either of two ways. It can be initiated as a Terminate and Stay Resident (TSR) module prior to creating or executing graphics software. Alternatively, it can be linked in as a component of the programmer's own CGP development project and used for batch printing. PrintImage provides options for rotating the image and increasing the size of the image when printed. See Part V for a complete discussion of image printing.

The MS-DOS and PC-DOS microcomputer operating system packages also supply an image printing program. This program, called GRAPHICS.COM, only functions as a TSR and has virtually no flexibility. Interested readers should refer to their DOS user manuals for more information.

2.3. Specifications

KPS routines that replace GU routines are not described in complete detail. Readers who want more information, should refer to their TP Reference Guide (Borland 1988b). Most of the following sections will list items and provide a short description and possibly a simple example.

Several constants that effect the "tuning" of the package, may be changed as the programmer sees fit. These and other miscellaneous Pascal TYPES and predeclared CGP variables are listed below.

CONSTANTS:

MaxPolySize = 1000
A polygon may not have more than MaxPolySize / 2 vertices.

MaxNumViewports = 10
At most MaxNumViewports viewports may be defined at one time.

MaxNumWorlds = 10
At most MaxNumWorlds world coordinate systems may be defined at one time.

MaxErrorText = 20
No more than MaxErrorText error messages will be retained.

The following Pascal TYPE corresponds to the **ArcCoordsType** defined in the GU, but the screen locations are in world coordinates.

TYPES:

```
NArcCoordsType = RECORD
  X,Y,
  XStart, YStart,
  XEnd, YEnd            : REAL;
End;
```

The following two variables maintain the current list of error messages (see section 2.3.2).

VARS:

```
NErrors : WORD;
ErrorText : Array[1..MaxErrorText] of STRING;
```

2.3.1. Dependencies

To compile the KPS, the TP unit Graph, and the CGP unit CGPHCS (HCS), must be present in the current directory or the TP unit directory. Furthermore, any TP font files and graphics driver files required by the programmer's own program must be available when the program is run, although they may be in a different directory.

2.3.2. Error handling services

An error messaging service is provided to record non-fatal error conditions. There are three support routines and two variables exported by the CGP to allow the programmer to examine, at any time, the current "stack" of error messages. It is important to note that messages can only be displayed on the text screen. If the program is in graphics mode, it must switch into text mode prior to printing (to the screen) the error message stack.

Declarations: **Procedure RecordError(ErrStr : STRING);**
Procedure ResetError;
Procedure PrintError;

Parameters: **ErrStr** - An arbitrary text string, usually describing an error condition.

Globals: **NErrors** - Indicates the current number of recorded messages.

Remarks: **RecordError** is used internally by the CGP and may be used by the programmer to stack his own error messages. The procedure **ResetError**, will set the variable **NErrors** to zero (**NErrors** greater than zero indicates messages are on the stack). By testing **NErrors** for a value greater than zero, the programmer can determine when to call **PrintError** to display the **NErrors** messages. In addition to displaying the error message stack, **PrintError** calls **ResetError** to set the message stack to empty.

2.3.3. Initialization and Loading services

Before any graphics oriented actions are taken, the GU must be notified of the type of graphics driver to use, the resolution/color mode, and where to find driver and stroked-font files. The CGP supplies two routines to accomplish this, one in the KPS and another in the BPS. If only services from the KPS are to be referenced, **InitKPS** should be called; if the BPS or higher level services are to be used, **InitCGP** should be invoked. Additionally, two routines are supplied for toggling hard copy printing on and off (currently not supported).

INITIALIZE GRAPHICS:

Declaration: **Procedure InitKPS(VAR GraphDriver,GraphMode : INTEGER;**
DriverPath : STRING);

Parameters: **GraphDriver** - Specifies which graphics driver to load. Driver name constants (below) are exported by the GU (Borland 1988b, p. 290).

INPUT:
Detect: match a driver to system hardware
or **CGA, MCGA, EGA, EGA64, EGAMono, Herc, ATT400, VGA, PC3270, 8514**

OUTPUT:
if successful: current graphics driver number
or **-2**: Cannot detect a graphics card
-3: Cannot find driver file
-4: Invalid driver
-5: Insufficient memory to load driver
-10: Invalid graphics mode for selected driver
-15: Invalid device number

GraphMode - Specifies which graphics mode to initiate for the requested driver. Mode name constants (below) are exported by the GU (Borland 1988b, p. 290). Each mode name is prefixed by its associated driver name.

INPUT:
0: match highest mode capability to system hardware
or **CGAC0, CGAC1, CGAC2, CGAC3, CGAHi, MCGAC0, MCGAC1, MCGAC2, MCGAC3, MCGAMed, MCGAHi, EGALo, EGAHi, EGA64Lo, EGA64Hi, EGAMonoHi, HercMonoHi, ATT400C0, ATT400C1, ATT400C2, ATT400C3, ATT400Med, ATT400Hi, VGALo, VGAMed, VGAHi, PC3270Hi, IBM8514Lo, IBM8514Hi**

OUTPUT:
current graphics mode for current driver

DriverPath - DOS disk/directory path where BGI driver files reside.

Remarks: This, or InitCGP in the BPS, must be called before any other KPS routines are referenced. It clears the screen, sets up the whole screen as one screen region, and establishes the default viewport and world coordinate system (see sections 1.6 and 1.7). For more information see Borland (1988b, pp. 290, 313).

LOAD DRIVER FILE:

Declaration: **Function LoadDriver(FileString : STRING) : BOOLEAN;**

Parameters: **FileString** - DOS disk/directory path where BGI driver files reside.

Remarks: This routine forces loading of a driver file from the **FileString** file and path. If one of the initialization routines is called with a driver path, this isn't necessary. If it is used, it must be called before an initialization routine. It returns TRUE if the driver is successfully loaded. For more information, see the description of the TP GU routine **RegisterBGIdriver** (Borland 1988b, p. 357).

LOAD STROKED-FONT FILE:

Declaration: **Function LoadFont(FileString : STRING) : BOOLEAN;**

Parameters: **FileString** - DOS disk/directory path where CHR stroked-font files reside.

Remarks: To load a stroked-font file, this routine should be called (the default bit-mapped font is always present). The BPS service **MapFont** calls this routine to satisfy font requests. **LoadFont** returns TRUE if the font was successfully loaded. Using this routine allows several stroked-fonts to co-exist in memory, providing quick font access. Font loading is automatic when BPS routines are used. For more information, see the description of the TP GU service **RegisterBGIfont** (Borland 1988b, p. 359).

HARD COPY:

Declaration: **Procedure HardCopyOn;**
Procedure HardCopyOff;

Remarks: These turn on and off the calls to HCS routines. In the current package, these have no effect.

2.3.4. Region, Viewport, World support services

DEFINE REGION:

Declaration: **Procedure DefineRegion(Nx,Ny : WORD);**

Parameters:

Nx - Specifies screen divisions into left and right halves.
May use these values:
1: no division
2: divide into left and right halves

Ny - Specifies screen divisions into top and bottom halves.
May use these values:
1: no division
2: divide into top and bottom halves

Remarks: Use this routine to set up one, two, or four independent drawing regions on the screen. The screen is cleared, world 0, viewport 0, and region 1 are selected.

NEXT REGION:

Declaration: **Procedure NextRegion;**

Remarks: On each successive call to **NextRegion**, the next drawing region is made active. Cycling through regions occurs from left to right, then top to bottom, selecting the first region if **NextRegion** is called while the last region is selected.

DEFINE VIEWPORT:

Declaration: **Procedure DefineViewport(I : INTEGER;
Rx1,Ry1,Rx2,Ry2 : REAL);**

Remarks: Given I in the range 1 to **MaxNumViewports**, viewport I is defined with left corner **(Rx1,Ry1)** and right corner **(Rx2,Ry2)**, with values in the range 0.0 to 1.0, relative to the active screen region. The definition remains in effect until either redefinition or a call to an initialization routine is made.

SELECT VIEWPORT:

Declaration: **Function SelectViewport(VAR I : INTEGER) : BOOLEAN;**

Remarks: Selects viewport I if it has been defined. Returns TRUE if that viewport is defined, FALSE otherwise. The selection remains in effect until another is selected or an initialization routine is executed.

DRAW VIEWPORT BOX:

Declaration: **Procedure DrawVPBox;**

Remarks: Draws a box (border) around the current viewport using the current line style, color, pattern, and thickness.

CLEAR VIEWPORT:

Declaration: **Procedure NClearViewport;**

Remarks: Erases the screen contents of the current viewport.

DEFINE WORLD:

Declaration: **Procedure DefineWorld(I : INTEGER;**
Rx1,Ry1,Rx2,Ry2 : REAL);

Remarks: Defines a world coordinate system identified by world I which must be in the range 1 to **MaxNumWorlds**. The coordinates are lower-left corner (**Rx1,Ry1**) and upper-right corner (**Rx2,Ry2**). The definition remains in effect until redefinition or execution of an initialization routine.

SELECT WORLD:

Declaration: **Function SelectWorld(I : INTEGER) : BOOLEAN;**

Remarks: Selects world coordinate system I, which remains in effect until another is selected or an initialization routine is executed. Returns TRUE if the world is defined, FALSE otherwise.

CLIPPING:

Declaration:	**Procedure TurnClipOn;** **Procedure TurnClipOff;**
Remarks:	Turn clipping on and off for subsequent drawing actions.

2.3.5. Graphics cursor movement

CURSOR POSITION READBACK:

Declaration:	**Function NGetX : REAL;** **Function NGetY : REAL;**
Remarks:	Returns the current X and Y graphics cursor coordinate in world coordinates. This depends on the active viewport and world.

CURSOR MOVEMENT:

Declaration:	**Procedure NMoveTo(Rx,Ry : REAL);** **Procedure NMoveRel(Rdx,Rdy : REAL);**
Remarks:	**NMoveTo** positions the graphics cursor at the world coordinate (**Rx,Ry**). **NMoveRel** moves a relative distance from the current cursor position (x,y) to a new position (**x+Rdx,y+Rdy**).

2.3.6. Point oriented services

WRITE/READ PIXEL:

Declaration:	**Procedure NPutPixel(Rx,Ry : REAL; Pixel : WORD);** **Function NGetPixel(Rx,Ry : REAL) : WORD;**
Parameters:	**Pixel** - specifies a color value (0..15). May use these color name constants:

Black = 0
Blue = 1
Green = 2
Cyan = 3
Red = 4
Magenta = 5
Brown = 6
LightGray = 7
DarkGray = 8
LightBlue = 9
LightGreen = 10
LightCyan = 11
LightRed = 12
LightMagenta = 13
Yellow = 14
White = 15

Remarks: The TP GU supplies **PutPixel** and **GetPixel** that perform a similar function except that they require the use of physical pixel locations rather than the world coordinates used here. The variable **Pixel** in **NPutPixel** is a number representing a drawing color. **NGetPixel** returns a color value in the range 0 to 15 (see **Pixel** above). Refer to the TP Reference Guide (Borland 1988b, pp. 300, 351) for more details.

WORLD PIXEL SIZE:

Declaration:

```
Function XSizeOfPixel : REAL;
Function YSizeOfPixel : REAL;
```

Remarks: These return the size of one pixel in the X and Y directions. Moving one pixel in the positive X direction is equivalent to incrementing the world system X coordinate by **XSizeOfPixel**. For example if the world coordinate system X axis ranges from -10.0 to 5.0, and the screen contains 320 pixels in the horizontal direction, and the current viewport covers half of the horizontal extent of the screen, **XSizeOfPixel** would return the value 15.0 / 160 = 0.094. A similar definition applies to **YSizeOfPixel**.

2.3.7. Line drawing services

LINE DRAWING:

Declaration:

```
Procedure NLine(Rx1,Ry1,Rx2,Ry2 : REAL);
Procedure NLineTo(Rx,Ry : REAL);
Procedure NLineRel(Rdx,Rdy : REAL);
```

Remarks: All three routines work with world coordinates. **NLine** draws a line segment from (**Rx1**,**Ry1**) to (**Rx2**,**Ry2**). **NLineTo** draws a line segment from the current position (x,y) to (**Rx**,**Ry**). **NLineRel** draws a line from the current position (x,y) a relative distance to (x+**Rdx**,y+**Rdy**). All three routines draw using the current color, style, and pattern. **NLine** draws lines in the current line thickness. Refer to the TP Reference Guide (Borland 1988b, pp. 325-327) for more details.

SET LINE STYLE:

Declaration: **Procedure NSetLineStyle(LineStyle,Pattern, Thickness : WORD);**

Parameters: **LineStyle** - may be one of:

SolidLn = 0
DottedLn = 1
CenterLn = 2
DashedLn = 3
UserBitLn = 4

Pattern - specifies a two byte rotating line pattern. It is ignored unless **LineStyle** equals **UserBitLn**.

Thickness - may be one of:

NormWidth = 1
ThickWidth = 3

Remarks: Note that there is a bug in the GU such that **LineTo** does not use the **Thickness** parameter. In anticipation of this bug being fixed sometime in the future, no fix has been written into the CGP routine **NLineTo**. Use **NLine** when line width control is desired. Refer to the TP Reference Guide (Borland 1988b, p. 385) for additional information.

2.3.8. Polygons and Shape filling services

These six CGP services use world coordinates. The equivalent GU services use physical pixel locations.

RECTANGLE:

Declaration: **Procedure NRectangle(Rx1,Ry1,Rx2,Ry2 : REAL);**

Remarks: Draw a rectangle with lower left corner **(Rx1,Ry1)** and top right corner **(Rx2,Ry2)** using the current color and line style.

BARS:

Declaration:
Procedure NBar(Rx1,Ry1,Rx2,Ry2 : REAL);
Procedure NBar3D(Rx1,Ry1,Rx2,Ry2,Depth : REAL; Top : BOOLEAN);

Remarks: **NBar** is simply a rectangle filled using the current fill style. **NBar3D** draws a three-dimensional rectangle, filled in the current fill style with an outline using the current line style and color with the specified **Depth** (world coordinates). **Top** passed as **TRUE** indicates the required presence of a three-dimensional top.

POLYGONS:

Declaration: **Procedure NDrawPoly(NumPoints : WORD; VAR RPolyPoints);**
Procedure NFillPoly(NumPoints : WORD; VAR RPolyPoints);

Remarks: **NDrawPoly** draws a polygon of **NumPoints** with the data points specified by **RPolyPoints** using the current color and line style. **NFillpoly** draws a polygon as **NDrawPoly** does, but also fills it using the current fill style.

SHAPE FILL:

Declaration: **Procedure NFloodFill(Rx,Ry : REAL; Border : WORD);**

Parameters: **Border** - Specifies a border color in the range 0-15. See **NPutPixel** above for a list of valid color name constants.

Remarks: Fills a region, using the current fill style, bounded by the color **Border**. If the seed point, **(Rx,Ry)**, is inside the shape, then the inside is filled, otherwise the outside is filled.

2.3.9. Circle, Arc, and Curve services

NCircle, **NArc**, and **NPieSlice** produce shapes which are circular as seen on the screen, and thus may not be circles in world coordinates. The radius is specified in units of the world X coordinate.

ARCS:

Declaration: **Procedure NArc(Rx,Ry,Radius : REAL;**
StAngle,EndAngle : WORD);
Procedure NGetArcCoords(VAR AC : NArcCoordsType);

Remarks: **NArc** produces a circular arc around **StAngle** to **EndAngle** with a radius **Radius** around the point **(Rx,Ry)** using the current line style, pattern, and color. **NGetArcCoords** retrieves information about the last **NArc** command.

CIRCLES:

Declaration:	**Procedure NCircle(Rx,Ry,Radius : REAL);**
Remarks:	Draws a circle with radius **Radius** and (**Rx**,**Ry**) as the center point using the current line style, pattern, and color.

ELLIPSES:

Declaration:	**Procedure NEllipse(Rx,Ry,Xr,Yr : REAL;** **StAngle,EndAngle : WORD);**
Remarks:	Draws an elliptical arc from **StAngle** to **EndAngle** with (**Rx**,**Ry**) as the center point using the current line style, pattern, and color. **Xr** is the X radius and **Yr** is the Y radius, both in world coordinates.

PIE SLICES:

Declaration:	**Procedure NPieSlice(Rx,Ry,Radius : REAL;** **StAngle,EndAngle : WORD);**
Remarks:	Draws and fills a pie slice with (**Rx**,**Ry**) as the center point, drawing from **StAngle** to **EndAngle** with a radius **Radius**, in the current line style, pattern, and color.

2.3.10. Text writing services

TEXT OUTPUT:

Declaration:	**Procedure NOutText(TextString : STRING);** **Procedure NOutTextXY(Rx,Ry : REAL; TextString : STRING);**
Remarks:	**NOutText** writes **TextString** at the current cursor position using the current font, color, orientation, justification, and size. **NOutTextXY** is similar to **NOutText** except it writes the string at position (**Rx**,**Ry**) without updating the current cursor position.

CHARACTER SIZE:

Declaration:	**Procedure NSetUserCharSize(FactorX,FactorY : REAL);**
Remarks:	**NSetUserCharSize** establishes scaling factors, which must be greater than zero, for stroked-fonts in the X and Y directions.

GET TEXT SIZE:

Declaraction:	**Function NTextHeight(TextString : STRING) : REAL;** **Function NTextWidth(TextString : STRING) : REAL;**
Remarks:	These return the height and width in world coordinates of **TextString**. Note that the services **CharHeight**, **CharDepth**, and **StringWidth** in the BPS are of greater utility for most users.

Chapter 3 - BASIC PLOTTING SERVICES

This unit, which we refer to as the BPS, provides many constants, types, variables, procedures, and functions that are used for high level graphing. It is essentially a support unit for the 2DPS (chapter 4) and 3DPS (chapter 5) units. To use this unit in a program, include a reference to the CGPBPS in the program's USES statement, together with any other required unit references.

Service names and parameter lists	Page no.
InitCGP(VAR GraphDriver,GraphMode : INTEGER; DriverPath : STRING);	29
Axis(AxisName : WORD; Min,Max : REAL; OptionStr,LabelStr : STRING);	30
AxisD(NpCats : WORD; VAR CatLabels; LabelStr : STRING);	31
Transform(N : WORD; VAR InArray,OutArray; AxisName : WORD);	32
MaxInArray(N : WORD; VAR A) : REAL;	32
MinInArray(N : WORD; VAR A) : REAL;	32
Log10(X : REAL) : REAL;	32
StripBlanks(InString : STRING) : STRING;	33
MapFont(FontUse,FontName : WORD; FactorWidth,FactorHeight : REAL);	33
OutAText(TextStr : STRING);	35
OutATextXY(X,Y : REAL; TextStr : STRING);	35
SetTextAlignment(HorizAlign : HorizAlignType; VertAlign : VertAlignType);	35
SetFontAndDirection(FontUse,Direction : WORD);	36
CharHeight(FontUse,Direction : WORD) : REAL;	36
CharDepth(FontUse,Direction : WORD) : REAL;	36
StringWidth(FontUse,Direction : WORD; TextString : STRING) : REAL;	37
MainHeader(HString : STRING);	37
SubHeader(HString : STRING);	37
SetSymbolSize(SymbolSizeFactor : REAL);	37
PlaceSymbol(X,Y : REAL; Symbol : WORD);	38
OuterFrame;	38

3.1. Dependencies

One TP unit and two CGP units are required by the BPS. These are the GU, KPS, and HCS. In addition to unit dependencies, the BPS requires that TP graphics driver files and stroked-font files are available when a program using the BPS routines is run.

3.2. Missing data

When a data point passed to a plotting routine has a value which is greater than or equal to **MissingTestValue**, it will not be plotted. This constant has a different value depending on the availability of MATH chip support. Version 5 of Turbo Pascal provides MATH chip emulation services (Borland 1988a, p. 177). If this option is enabled, PC's without a MATH chip will store REAL numbers in the same format and size as PC's with a MATH chip.

With Intel Math Co-processor (80x87):
MissingDataValue = 3.4E38
MissingTestValue = 3.3999E38

Without Intel Math Co-processor (80x87):
MissingDataValue = 1.0001E38
MissingTestValue = 1.0E38

3.3. Label sizes

Arrays of label strings passed to, for example, **PlotBarS** in the 2DPS, must conform to this type. **AxisD**, in the BPS, accepts a parameter **CatLabels** which is also an array of **LabelString**.

LabelString = STRING[30]

3.4. Plot box color control

FieldColor, BoxColor, LabelColor, VPBoxColor : WORD

FieldColor is initialized to the background color. The final three variables are initialized to the default drawing color. These may be changed by the programmer. Labelling text (but not header or sub-header text) is drawn in **LabelColor**. **BoxColor** is used for drawing the box, if one is specified, around the actual plot area, including tick marks, axes, and grids. **FieldColor** is the background color within the plot area. **VPBoxColor** is the color of the outer viewport box (border).

3.5. Initialization services

Declaration:	**Procedure InitCGP(VAR GraphDriver,GraphMode : INTEGER; DriverPath : STRING);**
Remarks:	This routine supercedes the KPS routine InitKPS and performs some additional initialization required by the BPS. It needn't be called unless the BPS or higher level units are used. See InitKPS in section 2.3.3 for more details.

3.6. Axis services

There are two kinds of axis services provided, one for continuous variables and the other for discrete variables. These services do not perform any actual graphics actions, they simply setup some internal configuration details to be used by plotting services in the 2DPS and 3DPS.

CONTINUOUS VARIABLE AXIS:

Declaration: **Procedure Axis(AxisName : WORD; Min,Max : REAL; OptionStr,LabelStr : STRING);**

Parameters: **AxisName** - Specifies the name of the axis to configure. May use these constants:

XAxis = 1:	Continuous variable X axis.
YAxis = 2:	Primary Y axis.
Y2Axis = 3:	Secondary Y axis.
ZAxis = 4:	Z axis (three-dimensional plots).
BarDataAxis = 1:	Bar graph data axis.

OptionStr - Provides a mechanism for specifying various plotting options (order not important). Each character specifies one option. The eight options supported are listed and described in Table 3.1 (p. 31).

Remarks: This routine must be called once for each continuous variable axis. When two dependent variable axes are used, the first is called **YAxis** and the second **Y2Axis**. The **YAxis** axis is drawn on the left side of the plot and the **Y2Axis** axis is drawn on the right. **Min** and **Max** are the minimum and maximum values of the variable. **LabelStr** is printed as the axis name (below or beside the axis).

Example: **Axis(XAxis,0.0,100.0,'BL','X Axis Label');**
The option string 'BL' indicates that the axis is to be logarithmic, and that major tick marks are to be placed on the bottom of the box.

Table 3.1 - List of Axis plotting options for continuous variables

Option code	Description
B	Place tick marks on the bottom (for **XAxis**) or left side (for **YAxis** or **Y2Axis**) of the box surrounding the plot region.
T	Place tick marks on the top or right side of the box.
M	Draw the minor tick marks.
R	Round the maximum and minimum values up and down to "nice" numbers.
D	Draw a solid line from **Min** to **Max** where the relevant variable is equal to 0.0.
G	Draw a dashed line grid at major tick values.
L	The axis is to be logarithmic.
P	The axis is to be a probability axis (see **Transform**, p. 32).

DISCRETE VARIABLE AXIS:

Declaration:

```
Procedure AxisD(NpCats : WORD; VAR CatLabels;
                         LabelStr : STRING);
```

Remarks: When the independent variable (X-axis) is discrete, **AxisD** must be called to specify the number of categories (**NpCats**), and the labels for these categories (**CatLabels**). These are passed in an array of **LabelString**, one array entry for each category. **LabelStr** will be printed as the axis name. This axis service is used with discrete plotting services provided by the 2DPS.

3.7. Data manipulation services

The services described in this section are tools for performing a number of data manipulations.

TRANSFORMATIONS:

Declaration: **Procedure Transform(N : WORD; VAR InArray,OutArray; AxisName : WORD);**

Parameters: **AxisName** - Specifies the name of the axis to associate the transformation with. May use these constants:

XAxis = 1: Continuous variable X axis.
YAxis = 2: Primary Y axis.
Y2Axis = 3: Secondary Y axis.
ZAxis = 4: Z axis (three-dimensional plots).

Remarks: If either the "L" or "P" option is passed to the **Axis** routine, **Transform** should be called to perform a logarithmic or probability (Abramowitz and Stegun 1965) scale transformation on the **N** elements of the array **InArray**, moving the results to **OutArray**. Both **InArray** and **OutArray** are arrays of an arbitrary length of type REAL. A call to **Axis** must occur before a call to **Transform**. If an **InArray** element is out of range for the specified transformation, that element is returned (untransformed) in **OutArray**.

MAXIMUM/MINIMUM:

Declaration: **Function MaxInArray(N : WORD; VAR A) : REAL;**
Function MinInArray(N : WORD; VAR A) : REAL;

Remarks: These routines return the largest (**MaxInArray**) or smallest (**MinInArray**) value of the **N** elements in the REAL array **A** (of arbitrary length).

LOGARITHM:

Declaration: **Function Log10(X : REAL) : REAL;**

Remarks: Given **X** > 0 and **X** < **MissingTestValue**, this function returns the logarithm to base 10 of **X**. Other values of **X** cause a value of zero to be returned.

STRIP BLANKS:

Declaration: **Function StripBlanks(InString : STRING) : STRING;**

Remarks: Given **InString**, this routine returns a string with all blank characters removed from **InString**.

3.8. Text services

There are several points the programmer should consider for text generation on the graphics screen. This section discusses text support functions, including font usage, text writing, text size, alignment, direction, and plot headers.

FONT MAPPING:

Declaration:

```
Procedure MapFont(FontUse,FontName : WORD;
                  FactorWidth,FactorHeight : REAL);
```

Parameters:

FontUse - Indicates which **FontUse** should be assigned **FontName** with width **FactorWidth** and height **FactorHeight**. See Table 3.2 (p. 34) for a list of valid font-use name constants.

FontName - Indicates which font to assign to **FontUse**. See Table 3.3 (p. 34) for a list of valid font name constants.

Remarks: Internally, the CGP determines the font to use at a given time by looking at a record corresponding to the intended use of the font. Calls to **MapFont** determine the correspondence between font use and font name (see Table 3.2, p. 34).

At initialization, all font uses are mapped onto the default font. **FactorWidth** and **FactorHeight** are real numbers by which the character width and height are to be multiplied. These factors must be greater than 0.0, but may be fractions. The actual character size which results depends upon the font specified in **FontName**, because the GU fonts have various default sizes. (The **DefaultFont** is drawn in an 8 x 8 pixel square, the **SmallFont** is 9 pixels high and the other fonts are 29 pixels high.) If **FontName** is **DefaultFont**, the smaller of **FactorWidth** and **FactorHeight** becomes the magnification factor, which may not be less than 1.0.

Fonts are loaded, as necessary, by **MapFont**. If no call to **FontPath** (see below) has been executed, the font files are expected to be in the current directory.

Example: **MapFont(AxisLabelFont,SmallFont,1.5,1.5)**
After this call, all axis labels will be drawn in the **SmallFont** with width and height magnifications of 1.5.

Table 3.2 - List of font-use name constants

Font use	Description
HeaderFont = 1	Used with main headers. See **MainHeader** routine.
AxisLabelFont = 2	Used with axis labels and subheaders.
TickLabelFont = 3	Used with tick mark numbers/labels.
PointLabelFont = 4	Used with data labels.
BarLabelFont = 5	Used with discrete variable (categories) labels.
LegendFont = 6	Used with legend text.

Table 3.3 - List of font name constants (scaled: 1.0 x 1.0)

Font name	Example text
DefaultFont	Default
TriplexFont	Triplex
SmallFont	Small
SansSerifFont	SansSerif
GothicFont	Gothic

Declaration: **Procedure FontPath(PathName : STRING);**

Remarks: When font files are not located in the current directory, this procedure must be called prior to any use of **MapFont** to specify the path to the font directory.

TEXT WRITING:

Declaration:

```
Procedure OutAText(TextStr : STRING);
Procedure OutATextXY(X,Y : REAL; TextStr : STRING);
```

Remarks: These two services output text in accordance with the layout specified by the routines **SetTextAlignment** and **SetFontAndDirection** (see below). **OutAText** writes text at the current cursor position, updating the cursor position accordingly, whereas **OutATextXY** writes text beginning at the point (**X**,**Y**) while leaving the cursor position unchanged.

TEXT ALIGNMENT:

Declaraction:

```
Procedure SetTextAlignment(HorizAlign : HorizAlignType;
                           VertAlign : VertAlignType);
```

Parameters:

HorizAlign - Specifies a horizontal position with respect to an (x,y) location for orienting text. May use these constants:

LeftTxt:	left justified
CenterTxt:	center on point
RightTxt:	right justified

VertAlignType - Specifies a vertical position with respect to an (x,y) location for orienting text. May use these constants:

TopTxt:	top
MiddleTxt:	between top and baseline
BaseLineTxt:	bottom of nondescenders
BottomTxt:	bottom of decenders

Remarks: The location of the alignment point within a text string is determined by **HorizAlign** and **VertAlign**. **HorizAlign** refers to the direction of the text, whether it is plotted vertically or horizontally. The alignment point may be at the left end (**LeftTxt**), center (**CenterTxt**) or at the right end (**RightTxt**). In the vertical direction (with respect to the text string, not the screen), the alignment point may be at the top (**TopTxt**), at the bottom of nondescending characters (**BaselineTxt**), halfway between the top and baseline (**MiddleTxt**), or at the bottom of descenders (**BottomTxt**).

FONT USE AND DIRECTION:

Declaration:	**Procedure SetFontAndDirection(FontUse,Direction : WORD);**
Parameters:	**FontUse** - Indicates which font use to assign **Direction** to. See Table 3.2 (p. 34) for a list of font-use name constants.
	Direction - Indicates which direction to orientate text to. May use these constants:

HorizDir	print horizontally
VertDir	print vertically

Remarks: This establishes the **FontUse** (e.g. **HeaderFont**) which will be used in subsequent calls to **OutAText** and **OutATextXY**, and determines the printing direction, which may be **HorizDir** or **VertDir**.

CHARACTER AND STRING SIZES:

Declaration:

CharHeight(FontUse,Direction : WORD) : REAL;
CharDepth(FontUse,Direction : WORD) : REAL;

Parameters:

FontUse - Indicates which font use to use for calculating the character height or depth. See Table 3.2 (p. 34) for a list of font-use name constants.

Direction - Indicates which direction to orientate text to for calculating character height or depth. May use these constants:

HorizDir	print horizontally
VertDir	print vertically

Remarks: For the specified **FontUse**, **CharHeight** returns the distance between text top and baseline and **CharDepth** returns descender size. The values returned are in world coordinate units, in the screen Y direction for **HorizDir** and in the screen X direction for **VertDir**.

Declaration: **Procedure StringWidth(FontUse,Direction : WORD; TextString : STRING) : REAL;**

Parameters: **FontUse** - Indicates which font use to use for calculating the string width for **TextString**. See Table 3.2 (p. 34) for a list of font-use name constants.

Direction - Indicates which direction to orientate text to for calculating the string width. May use these constants:

HorizDir	print horizontally
VertDir	print vertically

Remarks: Returns the world coordinate length of the given string, oriented as specified by FontUse and Direction.

PLOT HEADERS:

Declaration: **Procedure MainHeader(HString : STRING);**
Procedure SubHeader(HString : STRING);

Remarks: **MainHeader** causes HString to be drawn, in the font mapped to font-use **HeaderFont**, above the graph drawing area. **SubHeader** causes HString to be drawn below the main header in the font mapped to font-use **AxisLabelFont**. Header color may be established by a previous call to the GU routine SetColor.

3.9. Symbol services

SET SYMBOL SIZE:

Declaration: **Procedure SetSymbolSize(SymbolSizeFactor : REAL);**

Remarks: The default size for symbols is about 4 x 4 pixels, depending on aspect ratio. Following a call to this service they will be made larger or smaller by the factor specified.

DRAW SYMBOL:

Declaration: **Procedure PlaceSymbol(X,Y : REAL; Symbol : WORD);**

Parameters: **Symbol** - **Specifies which symbol shape to draw. Table 3.4 (p. 38) lists each symbol name constant with its corresponding numeric value and shape.**

Remarks: **The symbol shape specified by Symbol is drawn centered at the location (X,Y).**

Table 3.4 - List of plotting symbols

Symbol names and values	Symbols
PointSymbol = 0	·
PlusSymbol = 1	+
CrossSymbol = 2	×
OpenSquare = 3	□
OpenUpTriangle = 4	△
OpenDnTriangle = 5	▽
FilledSquare = 6	■
FilledUpTriangle = 7	▲
FilledDnTriangle = 8	▼
OpenCircle = 9	○
FilledCircle = 10	●
StarSymbol = 11	✱

3.10. Miscellaneous routines

SELECT OUTER FRAME:

Declaration: Procedure OuterFrame;

Remarks: Selects viewport 0 and world system 0.

Chapter 4 - TWO DIMENSIONAL PLOTTING SERVICES

This chapter discusses the two-dimensional high level services contained in the CGP unit 2DPS. To use this unit in a program, include a reference to the CGP2DPS in the program's USES statement together with any other required unit references.

Service names and parameter lists	Page no.
PlaneBox(Options : STRING; NLegend,NHeader,NyAxes : WORD);	43
PlotLine(N : WORD; VAR X,Y; YAxisName,LineStyle,LegendNum : WORD; LegendStr : STRING);	44
PlotSymbol(N : WORD; VAR X,Y; YAxisName,Symbol,LegendNum : WORD; LegendStr : STRING);	45
PlotBar(N : WORD; VAR X,Y; YAxisName,Pattern,LegendNum,TotBars,BarNum : WORD; LegendStr : STRING);	45
PlotSymbolD(VAR Data; BDT : BarDataType; Symbol,LegendNum : WORD; LegendStr : STRING);	46
PlotBarD(VAR Data; BDT : BarDataType; Pattern,LegendNum : WORD; LegendStr : STRING);	47
PlotBarS(VAR Data,SLabels,BarColors; BDT : BarDataType; Options : STRING);	47
AccumulateHistogram(VAR X,Y,A; N,Axis,Ints : WORD);	48
PlotHistoLine(NBin : WORD; VAR Y; YAxisName,LineStyle,LegendNum : WORD; LegendStr : STRING; LineOption : WORD);	49
PlotHistoSymbol(NBin : WORD; VAR Y; YAxisName,Symbol,LegendNum : WORD; LegendStr : STRING);	50
PlotHistoBar(NBin : WORD; VAR Y; YAxisName,Pattern,LegendNum : WORD; LegendStr : STRING);	50
LabelPoints(N : WORD; VAR X,Y,Labels; Location,YAxisName : WORD);	51
XErrorBar(N : WORD; VAR X1,X2,Y; T : REAL; YAxisName : WORD);	51
YErrorBar(N : WORD; VAR X,Y1,Y2; T : REAL; YAxisName : WORD);	51
Contour(Nx,Ny : WORD; VAR X,Y,Z; Nc : WORD; VAR Zc; Options : STRING; PosColor,ZeroColor,NegColor : WORD);	52

4.1. Overview

This unit deals with graphing in the plane, i.e., producing two-dimensional graphs. This does not include the projection of three-dimensional figures onto the plane, which is the subject of the 3DPS unit discussed in chapter 5. It does, however, include contour plotting.

Most users will find the 2DPS plotting capabilities easy to use, yet quite flexible, providing good control over graphics options. This unit provides the tools for quickly incorporating sophisticated graphic display capabilities into TP applications.

Shown in Figure 4.1 (p. 40) is a graph produced by executing the program given in Listing 4.1 (p. 41).

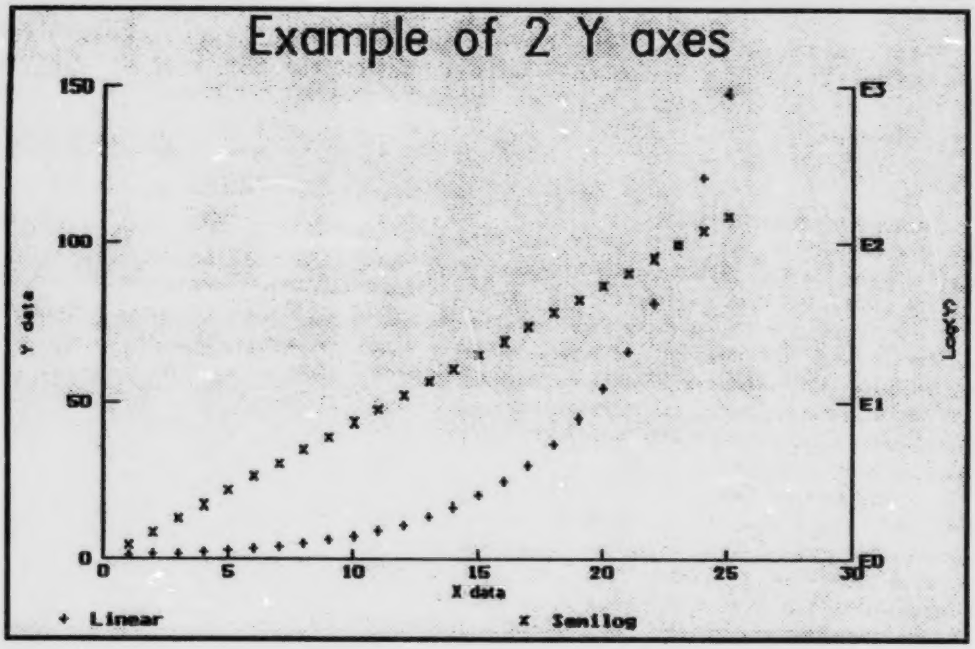


Figure 4.1 - Example of 2 y axes

Listing 4.1 - Example of 2 y axes

```
Program Fig4_1;

(* Requires TP units Crt and Graph plus two CGP units *)
Uses Crt, Graph, CGPBLU, CGPHLU2D;

Type R25 = Array[1..25] of REAL;
Var GraphDriver,GraphMode,i : INTEGER;  x,y : R25;  C : Char;

Begin
  (* --- Initialize graphics, autodetecting the graphics card --- *)
  GraphDriver := Detect; GraphMode := 0;
  InitCGP( GraphDriver,GraphMode,'C:\Turbo\Pascal4' );
  FontPath( 'C:\Turbo\Pascal4' );

  MapFont( HeaderFont,SansSerifFont,1.0,1.0 );
  MapFont( AxisLabelFont,SmallFont,1.0,1.0 );
  PlaneBox( 'BLRV',2,1,2 );

  (* --- Generate the example data --- *)
  For i := 1 to 25 do Begin  x[i] := i;  y[i] := Exp(i / 5);  End;

  (* --- Specify the X and Y axis --- *)
  Axis( XAxis,MinInArray(25,x),MaxInArray(25,x),'RB','X data' );
  Axis( YAxis,MinInArray(25,y),MaxInArray(25,y),'RB','Y data' );

  (* --- Draw plot box, axis, annotation etc. Plot (x,y) data points --- *)
  PlotSymbol( 25,x,y,YAxis,PlusSymbol,1,'Linear' );

  (* --- Specify the second Y axis with a Logarithmic transform --- *)
  Axis( Y2Axis,MinInArray(25,y),MaxInArray(25,y),'LTR','Log(Y)' );
  Transform( 25,y,y,Y2Axis );

  (* --- Plot (x,y2) data points --- *)
  PlotSymbol( 25,x,y,Y2Axis,CrossSymbol,2,'Semilog' );

  MainHeader( 'Example of 2 Y axes' );

  (* --- Pause for key press then return to text mode --- *)
  C := ReadKey;  CloseGraph;
End.
```

4.2. Specifications

Before any service with a name beginning with "Plot", or the point labelling or error bar services are referenced, the BPS service **Axis** (or **AxisD**) must be called for each axis to be used. Additionally, the 2DPS service **PlaneBox** must be called to set up the plotting area. These calls can be issued in any order. Once **Axis** has been called for any given plot axis, it remains in effect until overridden by another call referring to the same axis. Thus, multiple plots, with the same minimum, maximum, and options, may be made with only one call to **Axis**.

Procedure names ending with the letters "D" or "S" expect the independent axis to have been set up with a call to **AxisD** (discrete variables). The other plotting services work with continuous variables on both axes.

Data values passed to the plotting routines should be pretransformed. That is, if plotting is against a logarithmic or probability axis, the corresponding transformation should have been applied to the data. This may be accomplished by a call to the BPS routine **Transform**.

4.2.1. Dependencies

The 2DPS unit is dependent on the TP Graph unit and the CGP units BPS, KPS, and HCS. To recompile this unit, these four units must be accessible from the TP unit path (Borland 1988a, p. 184).

4.2.2. Plot limits

BAR CHART LIMITS:

A bar chart may contain no more than **MaxPCats** primary categories and **MaxSCats** secondary categories. The latter is limited by the number of distinct bar fill styles. In the procedure **PlotBar**, the minimum bar width is **MinBarWidthInPixels**, and this value will be used even if bar overlap occurs. These constant values are given below:

MaxPCats = 100
MaxSCats = 11
MinBarWidthInPixels = 5

LEGEND LABEL LIMIT:

When several variables are plotted on one graph, they may be distinguished by entries in a legend which is drawn at the bottom of the graph. This constant gives the largest allowable number of such entries.

MaxLegendNum = 12

CONTOUR LINE LIMIT:

The maximum number of distinguishable contour lines is given by:

MaxContourLineNdx = 10

4.2.3. PlaneBox

Declaration:	**Procedure PlaneBox(Options : STRING; NLegend,NHeader,NyAxes : WORD);**

Parameters:

Options - Specifies one or more options for setting up the plot area. See Table 4.1 (p. 44) for a list of valid options.

NyAxes - Specifies the number of Y-axes to reserve space for. May use these constants:

1	one Y axis on the left of the plot box
2	two Y axes one on the left and one on the right

Remarks:

This routine is used to specify layout details (i.e., reserve screen space) for all kinds of two-dimensional plots, whether of two continuous variables or one continuous and one discrete. It does not actually perform any graphics actions, it only initializes some internal configuration details to be used by the 2D plotting services. The options which are specified by single letters (in any order), are concatenated into the string **Options** (see Table 4.1, p. 44).

NLegend specifies the number of legend entries to leave space for at the bottom of the graph, with the value 0 indicating no legend strip. **NHeader** specifies the number of headers (0, 1, or 2). If **NHeader** = 1, there is to be a main header, while a value of 2 indicates that space should be left for a sub header as well. **NyAxes** indicates the number of Y axes (1 or 2) which will be used in drawing the plot. If one axis is specified, it will be drawn on the left side of the plot. If a second Y axis is used, it will be drawn on the right side. **NyAxes** = 2 has an effect only when the independent variable is continuous. The current viewport is cleared before plotting commences.

Table 4.1 - List of PlaneBox layout options

Option code	Description
B	Draw the bottom of the plot box.
T	Draw the top of the plot box.
L	Draw the left side of the plot box.
R	Draw the right side of the plot box.
D	The independent variable will be discrete.
H	Draw the plot with the discrete variable axis along the left edge. This has the effect of making bars horizontal. This option applies only if the independent variable is discrete.
V	Draw a box around the outer viewport in the color **VPBoxColor** (exported by the BPS).

4.2.4. Continuous - Continuous plotting

LINE PLOTS:

Declaration:

```
Procedure PlotLine(N : WORD; VAR X,Y;
                   YAxisName,LineStyle,LegendNum : WORD;
                   LegendStr : STRING);
```

Parameters:

YAxisName - Specifies which Y axis the plot applies to. May use these constants:

YAxis = 2:	Primary Y axis.
Y2Axis = 3:	Secondary Y axis.

LineStyle - may be one of:

SolidLn = 0
DottedLn = 1
CenterLn = 2
DashedLn = 3
UserBitLn = 4

Remarks: Draw a line connecting the **N** points at locations specified by the **REAL** arrays **X** and **Y**. **YAxisName** should be **YAxis** or **Y2Axis**, to indicate which dependent variable axis the **Y** array values are to be referred. Unless **LegendNum** = 0 or **NLegend** = 0 in the prior call to **PlaneBox**, the **LegendNum**'th entry in the legend will be drawn and labelled with **LegendStr**. If **UserBitLn** is the selected **LineStyle**, a call to **NSetLineStyle** (BPS) must preceed the call to **PlotLine**.

SYMBOL PLOTS:

Declaration:

```
Procedure PlotSymbol(N : WORD; VAR X,Y;
                     YAxisName,Symbol,LegendNum : WORD;
                     LegendStr : STRING);
```

Parameters: **YAxisName** - Specifies which Y axis the plot applies to. May use these constants:

YAxis = 2: Primary Y axis.
Y2Axis = 3: Secondary Y axis.

Symbol - Specifies which symbol shape to draw. Table 3.4 (p. 38) lists each symbol name constant with its corresponding numeric value and shape.

Remarks: Draw **Symbol** at each of the **N** locations specified by the **REAL** arrays **X** and **Y**. Refer to **PlotLine** for a description of the other parameters.

BAR CHARTS:

Declaration:

```
Procedure PlotBar(N : WORD; VAR X,Y; YAxisName,Pattern,
                  LegendNum,TotBars,BarNum : WORD;
                  LegendStr : STRING);
```

Parameters: **YAxisName** - Specifies which Y axis the plot applies to. May use these constants:

YAxis = 2: Primary Y axis.
Y2Axis = 3: Secondary Y axis.

Pattern - Specifies the fill pattern to use. Valid fill-pattern-name constants are listed in Table 4.2, p. 46.

Remarks: Draw **N** bars representing the data in the REAL arrays **X** and **Y**. The bar bottoms are on the y = 0 or y = y-minimum line, whichever is greater. If **TotBars** > 1, each bar is offset in the +- X direction from the corresponding data point so that **TotBars** calls may be made to **PlotBar** with the same **X** array, and the bars be placed side by side, centered over each X location. **BarNum** should be 1 for the first such call, and incremented by 1 for each subsequent call. **Pattern** is one of the eleven available GU fill patterns (Table 4.2, p. 46).

Table 4.2 - List of fill styles

Fill patterns and values		Patterns	
EmptyFill	**= 0**		**** Fill area in background color ****
SolidFill	**= 1**	■	**** Fill area in solid fill color ****
LineFill	**= 2**	≡	
LtSlashFill	**= 3**	////	
SlashFill	**= 4**	▨	
BkSlashFill	**= 5**	▧	
LtBkSlashFill	**= 6**	▧	
HatchFill	**= 7**	▦	
XHatchFill	**= 8**	◇◇◇	
InterleaveFill	**= 9**	▓	
WideDotFill	**= 10**	░	
CloseDotFill	**= 11**	▒	

4.2.5. Discrete - Continuous plotting

SYMBOL PLOTS:

Declaration: **Procedure PlotSymbolD(VAR Data; BDT : BarDataType;**
Symbol,LegendNum : WORD;
LegendStr : STRING);

Parameters: **BDT** - Specifies the data type used with the array **Data**. May use these constants:

WordType: array of type WORD
IntegerType: array of type INTEGER
RealType: array of type REAL

Symbol - Specifies which symbol shape to draw. Table 3.4 (p. 38) lists each symbol name constant with its corresponding numeric value and shape.

Remarks: Draw **Symbol** at the location specified by the continuous variable values specified in the array **Data**, one for each of the **NpCat** categories specified in a previous call to **AxisD**. The **Data** array may be of type WORD, INTEGER, or REAL, as indicated by the variable **BDT**. See section 4.2.4 (**PlotLine**, p. 44) for a description of the parameters **LegendNum** and **LegendStr**.

BAR CHARTS:

Declaration:

```
Procedure PlotBarD(VAR Data; BDT : BarDataType;
                   Pattern,LegendNum : WORD;
                   LegendStr : STRING);
```

Parameters: **BDT** - Specifies the data type used with the array **Data**. May use these constants:

WordType:	array of type WORD
IntegerType:	array of type INTEGER
RealType:	array of type REAL

Pattern - Specifies the fill pattern to use. Valid fill-pattern-name constants are listed in Table 4.2, p. 46.

Remarks: Draw **NpCat** (specified in a previous call to **AxisD**) bars with fill pattern **Pattern** and with heights specified by the values in the array **Data** (of type **BDT**). When **NLegend** > 1 in a previous call to **PlaneBox**, provision is made to place **NLegend** bars side by side for each of the **NpCat** categories. This is accomplished by repeated calls to **PlotBarD**, one for each secondary category.

STACKED BAR CHARTS:

Declaration:

```
Procedure PlotBarS(VAR Data,SLabels,BarColors;
                   BDT : BarDataType;
                   Options : STRING);
```

Parameters: **BarColors** - Each entry specifies a secondary category bar segment color. It is an array of WORD with **NLegend** (given in **PlaneBox**) entries. Each entry may use a GU color name constant (Borland 1988b, p. 406).

BDT - Specifies the data type used with the array **Data**. May use these constants:

WordType:	array of type WORD
IntegerType:	array of type INTEGER
RealType:	array of type REAL

Remarks: Draw stacked bars, one stack for each of the **NpCat** categories (specified in a previous call to **AxisD**), and with **NLegend** (given in a previous call to **PlaneBox**) segments in each stack. If **Options** = 'P', the bar stacks are each scaled to 100%. **BDT** specifies the data type of the array **Data**. **Data** is two-dimensional, with the first index varying from 1 to **NLegend** (number of secondary categories) and the second from 1 to **NpCat**. **SLabels** is an array of **LabelString** (see section 3.3), one element per secondary category.

4.2.6. Histogram plotting

ACCUMULATE HISTOGRAM:

Declaration: Procedure **AccumulateHistogram**(VAR **X**,**Y**,**A**; **N**,**Axis**,**Ints** : WORD);

Parameters: **Axis** - Identifies the bar data axis. May use this constant:

BarDataAxis

Remarks: Accumulate the **N** data points specified in the input REAL arrays **X** and **Y** into the output array **A**, dividing the axis specified by **Axis** into **Ints** equal intervals. For each X,Y pair, the value of X is used to determine which of the intervals is specified, and the Y value is accumulated into the corresponding element of **A**. The **Ints** elements of **A** are initialized to zero. This procedure prepares the data array Y for use by one of the following histogram routines.

For example if the **Axis** minimum value is 0.0 and its maximum value is 10.0, and Ints has the value 10, the data pair (3.5,6.2) would specify that the value 6.2 be added to the fourth element of array **A**.

LINE HISTOGRAMS:

Declaration:

```
Procedure PlotHistoLine(NBin : WORD; VAR Y; YAxisName,
                        LineStyle,LegendNum : WORD;
                        LegendStr : STRING;
                        LineOption : WORD);
```

Parameters:

YAxisName - Specifies which Y axis the plot applies to. May use these constants:

YAxis = 2:	Primary Y axis.
Y2Axis = 3:	Secondary Y axis.

LineStyle - may be one of:

SolidLn = 0
DottedLn = 1
CenterLn = 2
DashedLn = 3
UserBitLn = 4

LineOption - Specifies the type of histogram line segments to draw. May use these constants:

PointToPoint = 1:	straight line segments joining points.
Stepped = 2 :	conventional histogram stepped line.
StepBar = 3 :	vertical line at each interval boundary in addition to a conventional stepped line.

Remarks:

Plot a histogram with the X axis (extreme values previously specified in a call to **Axis** with **AxisName = BarDataAxis**) divided into **NBin** equal intervals. The **NBin** height values are given in the REAL array Y, and are to be referred to the Y axis **YAxisName**. A **LineOption** value of **PointToPoint** indicates that straight line segments are to be drawn joining the points given by the Y array values and X locations in the middle of each interval. When **LineOption** = **Stepped**, a conventional histogram stepped line is to be drawn, and when **LineOption** = **StepBar**, this stepped line is to be augmented with vertical lines from the X axis to the stepped line at each interval boundary. See section 4.2.4 (**PlotLine**, p. 44), for a description of **LegendNum** and **LegendStr**.

SYMBOL HISTOGRAMS:

Declaration:

```
Procedure PlotHistoSymbol(NBin : WORD; VAR Y; YAxisName,
                          Symbol,LegendNum : WORD;
                          LegendStr : STRING);
```

Parameters:

YAxisName - Specifies which Y axis the plot applies to. May use these constants:

YAxis = 2: Primary Y axis.
Y2Axis = 3: Secondary Y axis.

Symbol - Specifies which symbol shape to draw. Table 3.4 (p. 38) lists each symbol name constant with its corresponding numeric value and shape.

Remarks:

Place **NBin** symbols of type **Symbol** over the center of each X axis interval at the heights given by values in the REAL array Y. See PlotHistoLine (p. 49) for more details.

BAR HISTOGRAMS:

Declaration:

```
Procedure PlotHistoBar(NBin : WORD; VAR Y; YAxisName,
                       Pattern,LegendNum : WORD;
                       LegendStr : STRING);
```

Parameters:

YAxisName - Specifies which Y axis the plot applies to. May use these constants:

YAxis = 2: Primary Y axis.
Y2Axis = 3: Secondary Y axis.

Pattern - Specifies the fill pattern to use. Valid fill pattern name constants are listed in Table 4.2, p. 46.

Remarks:

Draw **NBin** bars with filled according to **Pattern**, one on each X axis interval with the heights given by the values in the REAL array **Y**. If **NLegend** was > 1 in a previous call to **PlaneBox**, provision is made for placing **NLegend** bars side by side on each X axis interval for successive calls to **PlotHistoBar**. See **PlotHistoLine** (p. 49) for more details.

4.2.7. LabelPoints

Declaration: **Procedure LabelPoints(N : WORD; VAR X,Y,Labels; Location, YAxisName : WORD);**

Parameters: **Location** - Specifies where to place text string labels relative to a data point. May use these constants:

RightOfPoint = 0:	right of the data point.
LeftOfPoint = 1:	left of the data point.
AbovePoint = 2:	above the data point.
BelowPoint = 3:	below the data point.
OnPoint = 4:	top of the data point.

YAxisName - Specifies which Y axis the plot applies to. May use these constants:

YAxis = 2:	Primary Y axis.
Y2Axis = 3:	Secondary Y axis.

Remarks: Draw the **N** text strings from the array **Labels** in position **Location** to the points specified in the REAL arrays **X** and **Y**. These label strings are written in the font which is mapped to **PointLabelFont**. The elements of **Labels** are of type **LabelString**, which is defined in the BPS (see section 3.3).

4.2.8. XErrorBar and YErrorBar

Declaration: **Procedure XErrorBar(N : WORD; VAR X1,X2,Y; T : REAL; YAxisName : WORD);**
Procedure YErrorBar(N : WORD; VAR X,Y1,Y2; T : REAL; YAxisName : WORD);

Parameters: **YAxisName** - Specifies which Y axis the plot applies to. May use these constants:

YAxis = 2:	Primary Y axis.
Y2Axis = 3:	Secondary Y axis.

Remarks: For **XErrorBar**, draw **N** horizontal error bars, at the heights given in the REAL array **Y**, between the X values given in the REAL arrays **X1** and **X2**. At the end of each bar draw a vertical line forming a tee with the error bar and having length **T**, where this length is specified as a fraction of the plot box height. (A useful value for **T** might be 0.01.) **YErrorBar** differs from **XErrorBar** in that the error bars are vertical as opposed to horizontal and the tee length is a fraction of the plot box length.

4.2.9. Contour

Declaration:

```
Procedure Contour(Nx,Ny : WORD; VAR X,Y,Z; Nc : WORD;
                  VAR Zc; Options : STRING;
                  PosColor,ZeroColor,NegColor : WORD);
```

Parameters:

Options - Specifies the contour plot options. See Table 4.3 (p. 53) for a valid list.

PosColor - If Options includes a 'C', this specifies the color for positive contours. See Borland (1988b, p. 406) for a list of valid color name constants.

ZeroColor - If Options includes a 'C', this specifies the color for zero contours.

NegColor - If Options includes a 'C', this specifies the color for negative contours.

Globals:

LinePatArray - If Options includes a 'Q', each entry (of Nc entries) specifies a 32-bit rotating line pattern for each contour level. Each bit equal to binary 1 represents a pixel.

LineColArray - If Options includes a 'D', each entry (of Nc entries) specifies a color for each contour level.

Remarks:

Produces a contour plot based in part on an algorithm by Bourke (1987). The **Nx** elements of the REAL array **X** and **Ny** elements of **Y** specify a rectangular grid (in which spacing need not be uniform). The two dimensional REAL array **Z** specifies a value for each intersection point. **Z**'s first index varies from 1 to **Nx** and its second from 1 to **Ny**. **Nc** specifies the number of contour levels and the **Nc** elements in the REAL array **Zc** give the values of each contour level. Before a call to this routine, as for other two-dimensional screen plotting routines, there must have been calls to **Axis** (one for the X axis and one for Y) and **PlaneBox**.

Contour lines may be distinguished by pattern, color or both. Patterns may be determined internally (option 'P') or specified externally (option 'Q') by setting values in the LongInt array **LinePatArray**. If the latter option is used, the programmer must set the values of the first **Nc** elements of **LinePatArray**. The 32-bit pattern specified by an array element is used cyclically, and values should be chosen that make continuous patterns when cycled.

Line colors may be specified in the procedure call (option 'C') or by setting elements in the array **LineColArray** (option 'D').

Table 4.3 - List of Contour option codes

Option codes	Description
T	Draw tick marks on the inside of the plot box to designate the grid line locations. To avoid confusion, when this option is used, the previous calls to **Axis** should specify no tick marks.
P	Distinguish the contour levels by line pattern. Contour lines at negative values will be drawn with a dashed line, those at zero will be dotted, and positive value contours will be solid lines.
Q	Use the elements in the array **LinePatArray** to specify contour line patterns. This option supersedes the 'P' option.
C	Distinguish the contour levels by color. The three parameters PosColor, Zerocolor, and NegColor will be used.
D	Use the elements of LineColArray to specify contour line color. This option supersedes the 'C' option.

Chapter 5 - THREE DIMENSIONAL PLOTTING SERVICES

This chapter discusses the three-dimensional high level routines exported by the CGP unit 3DPS. To use this unit in a program, include a reference to the CGP3DPS in the program's USES statement together with any other required unit references.

Service names and parameter lists	Page no.
CornerBox(Options : STRING; NLegend,NHeader : WORD);	59
Lolliplot(N : WORD; VAR X,Y,Z; Symbol,LegendNum : WORD; LegendStr : STRING);	59
LabelLolliPoints(N : WORD; VAR X,Y,Z,Labels; Location : WORD);	60
Surface(Nx,Ny : WORD; VAR X,Y,F; TopColor,BottomColor : WORD);	60

5.1. Overview

This unit deals with graphing three-dimensional objects in the plane. In order to view three-dimensional objects on a two-dimensional screen, we make a projection which "looks down" into the corner of a box specified by the X = constant, Y = constant, and Z = constant planes. Two types of graphs are supported, lollipop plots (3D scatter plots) and surface-view plots. This unit provides the tools for quickly incorporating sophisticated three-dimensional graphic display capabilities into TP applications.

Shown in Figure 5.1 is a graph produced by executing the program given in Listing 5.1 (p. 56).

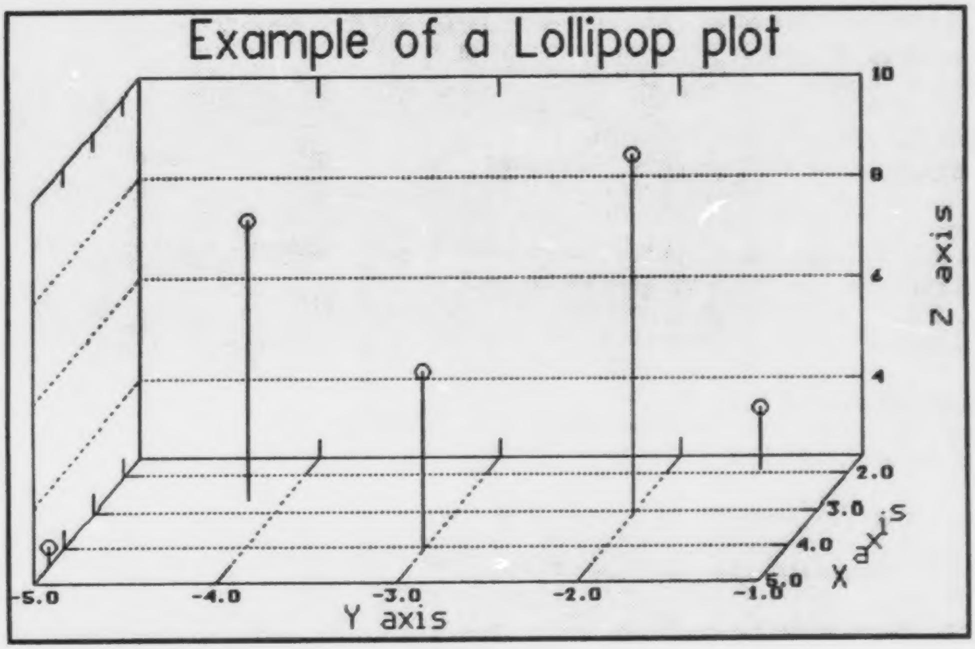


Figure 5.1 - Example of a lollipop plot

Listing 5.1 - Example of a lollipop plot

```
Program Fig5_1;

(* Requires TP units Crt and Graph plus two CGP units *)
Uses Crt, Graph, CGPBLU, CGPHLU3D;

Type R5 = Array[1..25] of REAL;
Var GraphDriver,GraphMode,i : INTEGER;  x,y,z : R5;  C : Char;

Begin
  (* --- Initialize graphics, autodetecting the graphics card --- *)
  GraphDriver := Detect; GraphMode := 0;
  InitCGP( GraphDriver,GraphMode,'C:\Turbo\Pascal4' );
  FontPath( 'C:\Turbo\Pascal4' );

  MapFont( HeaderFont,SansSerifFont,1.0,1.0 );
  MapFont( AxisLabelFont,SmallFont,1.5,1.5 );
  CornerBox( 'V',0,1 );

  MainHeader( 'Example of a Lollipop plot' );

  (* --- Generate the example data --- *)
  x[1] := 3.1; y[1] := -2.0; z[1] := 9.6;
  x[2] := 4.5; y[2] := -5.0; z[2] := 2.9;
  x[3] := 4.1; y[3] := -3.0; z[3] := 6.0;
  x[4] := 1.9; y[4] := -1.5; z[4] := 3.7;
  x[5] := 2.7; y[5] := -4.2; z[5] := 8.0;

  (* --- Specify the X, Y, and Z axes --- *)
  Axis( XAxis,MinInArray(5,x),MaxInArray(5,x),'RG','X axis' );
  Axis( YAxis,MinInArray(5,y),MaxInArray(5,y),'RG','Y axis' );
  Axis( ZAxis,MinInArray(5,z),MaxInArray(5,z),'RG','Z axis' );

  (* --- Specify symbol size and plot (x,y,z) data points --- *)
  SetSymbolSize( 1.5 );
  Lolliplot( 5,x,y,z,OpenCircle,0,' ' );

  (* --- Pause for key press then return to text mode --- *)
  C := ReadKey;  CloseGraph;
End.
```

As a second example, consider Figure 5.2 and the corresponding program in Listing 5.2 (p. 58).

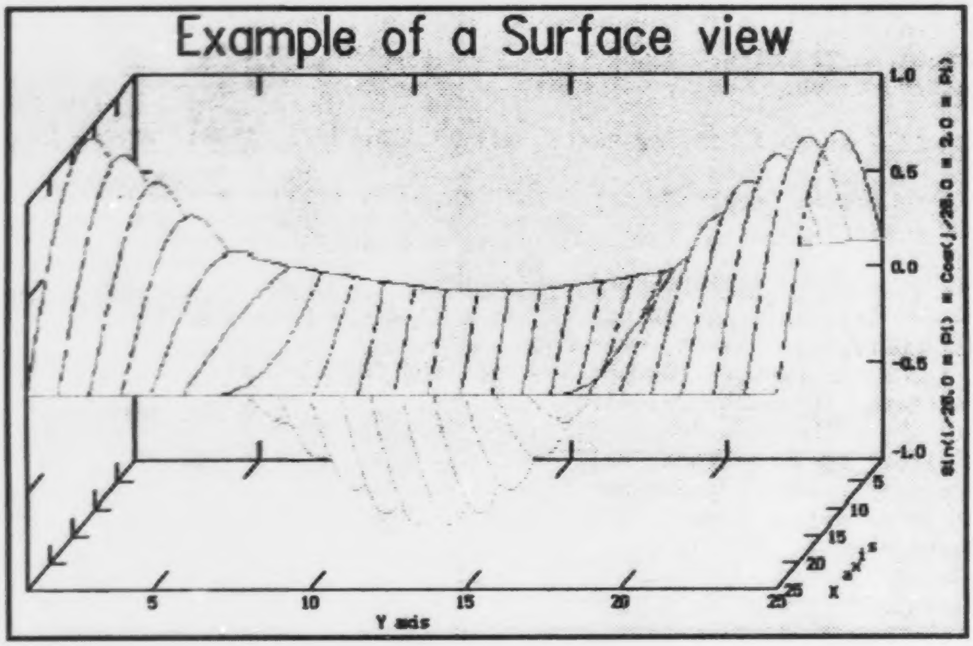


Figure 5.2 - Example of a surface plot

Listing 5.2 - Example of a surface plot

```
Program Fig5_2;

(* Requires TP units Crt and Graph plus two CGP units *)
Uses Crt, Graph, CGPBLU, CGPHLU3D;

Type R25 = Array[1..25] of REAL;  R25_25 = Array[1..25] of R25;

Var GraphDriver,GraphMode,i,j : INTEGER;  x,y : R25;  c : Char;
    F : R25_25;

Begin
  (* --- Initialize graphics, autodetecting the graphics card --- *)
  GraphDriver := Detect; GraphMode := 0;
  InitCGP( GraphDriver,GraphMode,'C:\Turbo\Pascal4' );
  FontPath( 'C:\Turbo\Pascal4' );

  MapFont( TickLabelFont,SmallFont,1.0,1.0 );
  MapFont( HeaderFont,SansSerifFont,1.0,1.0 );
  MapFont( AxisLabelFont,SmallFont,1.0,1.0 );
  CornerBox( 'V',0,1 );

  MainHeader( 'Example of a Surface view' );

  (* --- Generate the example data --- *)
  For i := 1 to 25 do x[i] := i;  For j := 1 to 25 do y[j] := j;
  For i := 1 to 25 do For J := 1 to 25 do
     F[i,j] := Sin(i/25.0 * Pi) * Cos(j / 25.0 * 2.0 * Pi);

  (* --- Specify the X, Y, and Z axes --- *)
  Axis( XAxis,MinInArray(25,x),MaxInArray(25,x),'','X axis' );
  Axis( YAxis,MinInArray(25,y),MaxInArray(25,y),'','Y axis' );
  Axis( ZAxis,-1.0,1.0,'','Sin(i/25.0 * Pi) * Cos(j/25.0 * 2.0 * Pi)' );

  (* --- Draw the surface --- *)
  Surface( 25,25,x,y,F,3,1 );

  (* --- Pause for key press then return to text mode --- *)
  C := ReadKey;  CloseGraph;
End.
```

5.2. Specifications

This unit is relatively small. It exports only four three-dimensional plotting routines. Before any of **Lolliplot**, **LabelLolliPoints**, or **Surface** is called, the BPS service **Axis** must be called for each of the X, Y, and Z axes.

5.2.1. Dependencies

The 3DPS unit is dependent on the TP Graph unit and the CGP units 2DPS, BPS, KPS, and HCS. To recompile this unit, all five units must be accessible from the TP unit path (Borland 1988a, p. 184).

5.2.2. CornerBox

Declaration:

```
Procedure CornerBox(Options : STRING;
                    NLegend,NHeader : WORD);
```

Parameters: **Options** - Specifies configuration options for 3D plots. See Table 5.1 for a list.

Remarks: This must be called before either the **Lolliplot** or **Surface** plotting routines. It does the basic plot layout calculations. **NLegend** specifies the number of legend entries to leave space for at the bottom of the graph, with the value 0 indicating no legend strip. **NHeader** specifies the number of headers (0, 1, or 2). If **NHeader** = 1, there is to be a main header, while a value of 2 indicates that space should be left for a subheader as well.

Table 5.1 - List of **CornerBox** layout options

Option code	Description
V	Draw a box around the outer viewport.
W	Draw the corner box with wide lines.

5.2.3. Lolliplot

Declaration:

```
Procedure Lolliplot(N : WORD; VAR X,Y,Z;
                    Symbol,LegendNum : WORD;
                    LegendStr : STRING);
```

Parameters: **Symbol** - Specifies which symbol shape to draw. Table 3.4 (p. 38) lists each symbol name constant with its corresponding numeric value and shape.

Remarks: Draw the **N** "lollipops" specified in the **REAL** arrays **X**, **Y**, and **Z** with **Symbol** forming the "candy". Label legend entry **LegendNum** with **LegendStr**. Essentially, this service produces a 3D scatter plot.

5.2.4. LabelLolliPoints

Declaration:

```
Procedure LabelLolliPoints(N : WORD; VAR X,Y,Z,Labels;
                                    Location : WORD);
```

Parameters: **Location** - Specifies where to place text string labels relative to a data point. May use these constants:

RightOfPoint = 0: right of the data point.
LeftOfPoint = 1: left of the data point.
AbovePoint = 2: above the data point.
BelowPoint = 3: below the data point.
OnPoint = 4: top of the data point.

Remarks: Draw the **N** character strings from the array **Labels** at the locations given by elements in the **REAL** arrays **X**, **Y**, and **Z**. The variable **Location** may take a point label placement value. The elements of **Label** are of type **LabelString**, defined in the BPS.

5.2.5. Surface

Declaration:

```
Procedure Surface(Nx,Ny : WORD; VAR X,Y,F;
                  TopColor,BottomColor : WORD);
```

Parameters: **TopColor** - Specifies a color for the visible top parts of the surface. See Borland (1988b, p. 406) for a list of color name constants.

BottomColor - Specifies a color for the visible bottom parts of the surface.

Remarks: Draw a view of the surface **F(X,Y)** specified by the three **REAL** arrays **X**, **Y**, and **F**. Color the visible top parts of the surface **TopColor** and the visible bottom parts of the surface **BottomColor**. The surface is represented as a series **Ny** lines at constant X, with hidden lines removed.

PART TWO

INTERACTIVE GRAPHICS SYSTEM (IGS)

Chapter 6 - IGS INTRODUCTION
Chapter 7 - IGS PROGRAMMER SERVICES
Chapter 8 - IGS USER MANUAL

Chapter 6 - IGS INTRODUCTION

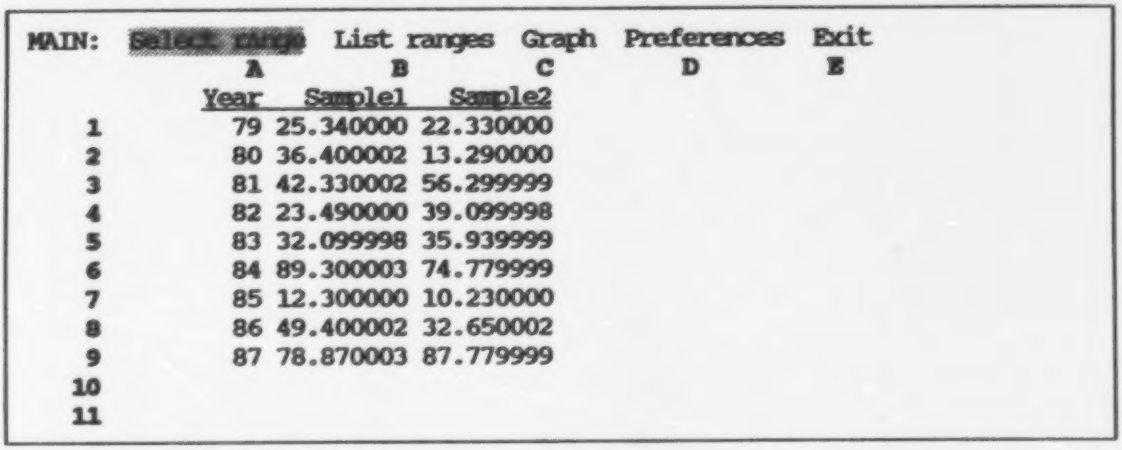
The Interactive Graphics System (IGS) is a menu-driven system for generating graphics on microcomputer display screens. It is based on a spreadsheet or worksheet concept of data with the data organized into a two-dimensional array in which the values in a given column share characteristics (title, presentation format, data type). Blocks of data (by row, by column, or mixed) are selected by defining ranges, and these blocks are passed to the plotting routines as variables.

To use the system, the programmer sets up certain arrays (including the data array) and passes pointers to these arrays via an entry point in the IGS. This invokes the menu system by which the user is able to select data ranges from the spreadsheet and direct the generation of graphs based on these ranges.

Graphics operations in the IGS are provided by the CGP (see Part I). The programmer using the IGS need not directly reference the CGP or the TP GU, but these units must be available for compiling and linking. Table 7.3 (p. 73) lists all of the required units.

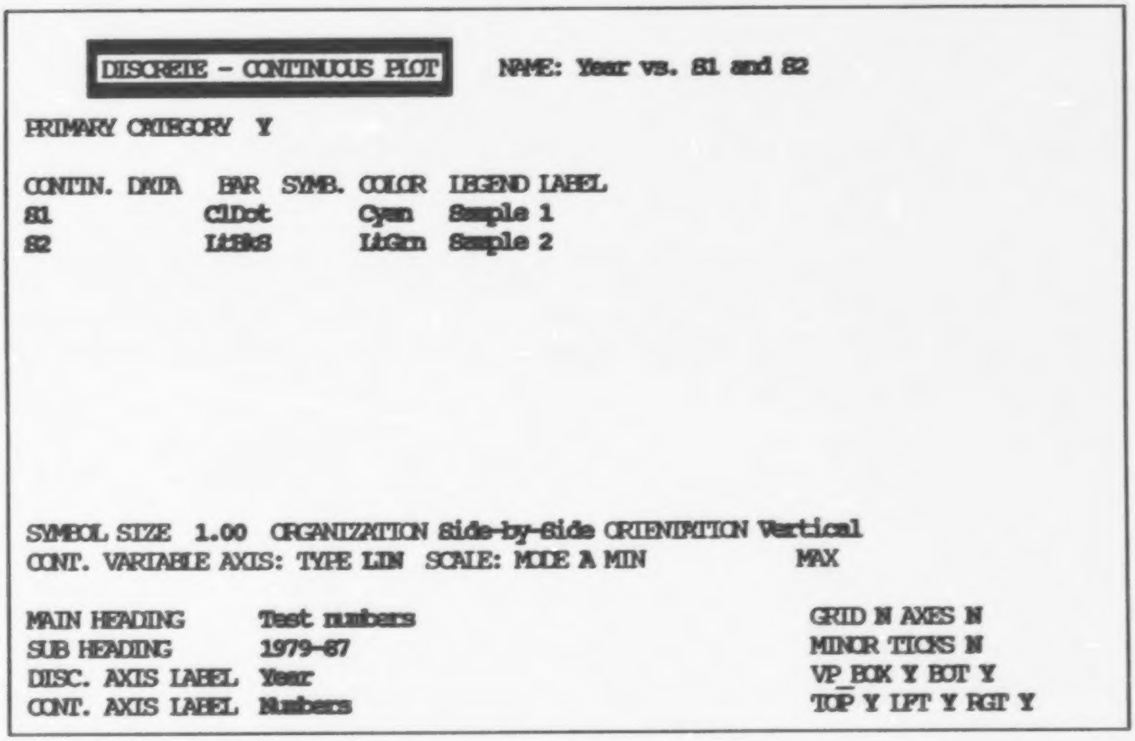
This document provides a programmer's reference manual (chapter seven), in which is specified the process of calling the IGS from a TP program, and a user manual (chapter eight), which describes the menu-driven user interface. Users intending to use only the CPP (see Part III) or the MRPGS (see Part IV) should read the IGS user manual in chapter eight.

The ensuing example demonstrates a typical data-set passed to the IGS by a simple TP program. The opening IGS screen and a graph setup screen are shown in Figures 6.1a-b (p. 63), and a typical graph is provided in Figure 6.1c (p. 64).



```
MAIN: Select range  List ranges  Graph  Preferences  Exit
                A          B          C          D          E
              Year   Sample1   Sample2
     1          79 25.340000 22.330000
     2          80 36.400002 13.290000
     3          81 42.330002 56.299999
     4          82 23.490000 39.099998
     5          83 32.099998 35.939999
     6          84 89.300003 74.779999
     7          85 12.300000 10.230000
     8          86 49.400002 32.650002
     9          87 78.870003 87.779999
    10
    11
```

Figure 6.1a - Example of the IGS opening screen



```
    DISCRETE - CONTINUOUS PLOT     NAME: Year vs. S1 and S2

PRIMARY CATEGORY  Y

CONTIN. DATA   BAR  SYMB. COLOR  LEGEND LABEL
S1             ClDot        Cyan   Sample 1
S2             LtBkS        LtGrn  Sample 2








SYMBOL SIZE  1.00  ORGANIZATION Side-by-Side ORIENTATION Vertical
CONT. VARIABLE AXIS: TYPE LIN  SCALE: MODE A MIN             MAX

MAIN HEADING        Test numbers                              GRID N AXES N
SUB HEADING         1979-87                                   MINOR TICKS N
DISC. AXIS LABEL  Year                                        VP_BOX Y BOT Y
CONT. AXIS LABEL  Numbers                                     TOP Y LFT Y RGT Y
```

Figure 6.1b - Example of an IGS graph setup screen

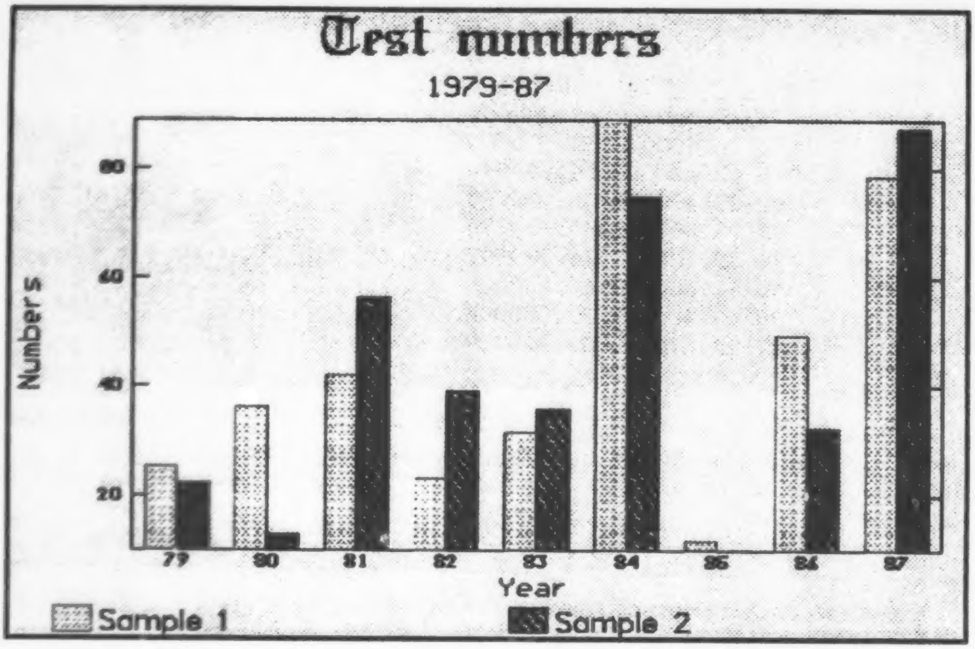


Figure 6.1c - Example of an IGS Discrete - Continuous graph

This example demonstrates only one of the plot types supported by the IGS. By selecting three ranges from the spreadsheet (Figure 6.1a, p. 63), and filling in the graph setup screen (Figure 6.1b, p. 63), the plot shown in Figure 6.1c is produced. Within minutes a variety of plots may be produced from the same data-set.

Chapter 7 - IGS PROGRAMMER SERVICES

This chapter should be read by TP programmers who wish to integrate the capabilities provided by the IGS into their own programs. Users of an IGS application should read chapter eight which discusses the workings of the user interface for the IGS.

Service names and parameter lists	Page no.
IGSMainMenu(DataArray : POINTER; Rows,Cols : INTEGER; ColTypeArray : DataFieldTypeArrayPtr; ColFmtArray : ColFmtArrayPtr; ColTitleArray : ColTitleArrayPtr);	66
IGSRelease;	69
SaveSession(SessionFilename : STRING);	69
LoadSession(SessionFilename : STRING) : BOOLEAN;	70
SavePrefs(Filename : STRING);	70
LoadPrefs(Filename : STRING);	70
StorePointerInReal(StringPtr : POINTER; VAR X : REAL);	70

7.1. Overview

The IGS consists of two TP units (IGSType and IGSMain) which user programs must reference and several support units (see Table 7.3, p. 73) which are used internally by the IGS. The IGSType unit contains a few Pascal constant and type declarations required for setting up a call to the IGS. This separate unit is required to avoid circularity of unit reference. IGSMain contains the remaining code through which TP programs interface with the IGS. A typical program that uses the IGS might read data from a text file, perform some transformation on the data, and then pass the data to the IGS for plotting service.

To use the IGS, the programmer must set up a data array of type REAL (without a MATH chip) or SINGLE (with a MATH chip), depending on how the IGS and its support units were compiled (i.e. with or without MATH chip support turned on). If version 5.0 of Turbo Pascal is used and a MATH chip isn't available, the MATH chip may be emulated. *Users should see the discussion of MATH chip support in section 1.1.*

The data array is two dimensional. Each column contains either real numbers or pointers to character strings, according to the specifications supplied in a programmer-supplied array of type **DataFieldTypeArray** (see section 7.3.1). A service is supplied to perform the operation of storing a pointer in a real number memory location.

Any real number array element with a value equal to or greater than **IGSMissingData**, will be interpreted as missing, and will appear as a blank field on the spreadsheet screen, and furthermore, will be ignored by the plotting routines. This variable is a real number which is nearly the largest which can be represented by whichever type (REAL or SINGLE) is in use.

The programmer may optionally provide an array of column titles and an array of format specifications for the spreadsheet. If these are not required, NIL pointers may be passed in place of these arrays.

Several services are provided to save and restore information about the system. Their use alleviates the need to reenter basic information each time the IGS is entered and allows re-use of named range definitions and plot definitions. Care must be taken by the IGS user that restored definitions suit the data array in use.

We will first document the main entry point to the IGS, then constants and types, then other support routines.

7.2. IGS Main Unit

7.2.1. IGSMainMenu

Declaration:

```
Procedure IGSMainMenu
            (DataArray : POINTER;
             Rows, Cols : INTEGER;
             ColTypeArray  : DataFieldTypeArrayPtr;
             ColFmtArray   : ColFmtArrayPtr;
             ColTitleArray : ColTitleArrayPtr);
```

Parameters:

ColTypeArray - Each element specifies the data type for each data column. There must be at least Cols elements in this array. Each element may point to these values:

NumericField:	column entries contain real numbers
LabelField:	column entries contain pointers to strings

ColFmtArray - Each element specifies the data format for each data column. Each element is subdivided into an 'M' part (field width) and an 'N' part (number of decimal places). Each element-subelement points to these integer values. A NIL pointer may be passed for this parameter to request default data formatting. See page 68 for more details.

ColTitleArray - Each element specifies a title for each data column. A NIL pointer may be passed for this parameter if titles are not required. Each element may point to strings ranging from 1 to 15 characters.

Remarks: This is the entry point to the IGS menu system. Note that most of the arguments are pointers. **Rows** and **Cols** specify, respectively, the number of rows and columns in the input data array to which **DataArray** points. This is an array of type REAL or SINGLE, as described in section 7.1. The elements in a given column of **DataArray^** will be interpreted, as specified by the elements of **ColTypeArray^**, as real numbers or pointers to strings.

When **ColFmtArray** is not NIL, it must point to an array of **ColFmtRec's** (see page 68), one element for each column. Refer to Table 7.1 (p. 69) for a list of possible **M** and **N** values for this array. When **ColTitleArray** is not NIL, it must point to an array of **ColTitle's** (see below), one for each column.

Example: IGSMainMenu(@Data,10,3,ColTypes,NIL,NIL);
Data is an array of REAL, dimensioned 10 rows by 3 columns. **ColTypes** is a pointer to an array of column types. Data formats and column titles will be set to the default settings.

CONSTANTS:

The spreadsheet array may have no more than **MaxCols** columns and the optional titles at the top of each column may contain no more than **MaxColTitleLen** characters each.

MaxColTitleLen = 15
MaxCols = 1000

TYPES:

Column data types:

DataFieldType = (**EmptyField**,**NumericField**,**LabelField**);
EmptyField is used internally only; **NumericField** indicates column entries are real numbers; **LabelField** indicates column entries are pointers to character strings.

DataFieldTypeArray = Array[1..MaxCols] of DataFieldType;
DataFieldTypeArrayPtr = ^DataFieldTypeArray;
Use this type to declare a list of column data types to pass to the IGS via the routine **IGSMainMenu**.

For example:

```
Var ColTypes : DataFieldTypeArrayPtr;
             .
             .
New( ColTypes );
             .
             .
ColTypes^[i] := NumericField;
             .
             .
```

<u>Column names</u>:

```
ColTitle = STRING[MaxColTitleLen];
WksColTitleArray = Array[1..MaxCols] of ColTitle;
ColTitleArrayPtr = ^WksColTitleArray;
```

Use this type to declare a list of column names to pass to the IGS via the routine **IGSMainMenu**.

For example:

```
Var ColNames : ColTitleArrayPtr;
             .
             .
New( ColNames );
             .
             .
ColNames^[i] := 'Column i';
```

<u>Column data formats</u>:

```
ColFmtRec = Record M,N : INTEGER; End;
ColFmtArray = Array[1..MaxCols] of ColFmtRec;
```

The record field 'M' specifies the column field width. The record field 'N' is described in Table 7.1.

```
ColFmtArrayPtr = ^ColFmtArray;
```

Use this type to declare a list of column data display formats to pass to the IGS via the routine **IGSMainMenu**.

For example:

```
Var ColFormats : ColFmtArrayPtr;
             .
             .
New( ColFormats );
             .
             .
ColFormats^[i].M := 10;       (* Field width *)
ColFormats^[i].N := 2;        (* Two decimal places *)
```

Table 7.1 - Column data format options

N value	Results
>= 0	Print numbers with N decimal places.
-1	Print numbers with floating point E format.
-2	Print numbers using free format.

7.2.2. IGSRelease

Declaration: **Procedure IGSRelease;**

Remarks: The IGS allocates memory for its internal use, including the storage of named range definitions and plot definitions. A call to this procedure causes all of this internal storage to be released, with consequent loss of the definitions. It should be called at the end of any IGS session, after the session definitions have been saved via **SaveSession**, if required.

7.2.3. SaveSession

Declaration: **Procedure SaveSession(SessionFilename : STRING);**

Remarks: This causes the writing to disk the current named range definitions and plot definitions. Definitions will be saved in files with path and name **SessionFilename** and internally specified extensions (.X00,.X01,.X02). If a null string is passed as the argument, **SaveSession** will prompt for the file name. If the required files do not exist, they will be created. Note that the data are not saved, only the session information; it is up to the programmer to provide the data to the IGS for each session.

Programmers can optionally set the variable **DataFileSpec** (of type STRING) equal to the name of the data file used in a program. This variable is written by **SaveSession** and read in by **LoadSession**. It may be used to save a file name from one session for use in another.

7.2.4. LoadSession

Declaration:	**Function LoadSession(SessionFilename : STRING) : BOOLEAN;**
Remarks:	This restores the session information saved to disk by **SaveSession**. It returns a value of TRUE if the restoration was successful, FALSE if not. It will prompt for the filename if passed a nul string. Note that only session information is loaded. The programmer must load the data arrays and provide the other required parameters to the IGS (IGSMainMenu).

7.2.5. SavePrefs

Declaration:	**Procedure SavePrefs(Filename : STRING);**
Remarks:	This writes to disk information about preferences, including that established by the Preferences screen from the main menu of the IGS and the Graph Options screen. It will not prompt for a file name.

7.2.6. LoadPrefs

Declaration:	**Procedure LoadPrefs(Filename : STRING);**
Remarks:	Restores from disk the information stored in a call to SavePrefs. It will not prompt for a file name. If the files does not exist, LoadPrefs simply returns to the caller.

7.2.7. StorePointerInReal

Declaration:	**Procedure StorePointerInReal(StringPtr : POINTER; VAR X : REAL);**
Remarks:	This routine may be used to store pointers into REAL array element locations. This is required for passing string pointers to IGSMainMenu in the spreadsheet array.

Example:

```
Var StrFlds : Array[1..5] of STRING;

Begin
               .

               .

  For j := 1 to 5 do
   Begin
      Write( 'Enter string: ' ); ReadLn( StrFlds[j] );
      StorePointerInReal( @StrFlds[j],Data[1,j] );
   End;
               .

               .

End.
```

7.3. An example program

The program in Listing 7.1 (p. 72) simply reads in the data file given in Table 7.2 (p. 71). Column one is a list of years. The years are treated as character strings rather than numbers, therefore the program places pointers to the strings in the data array for this column.

The program sets up column titles in the variable ColNames. Examining Figure 6.1a (p. 63) shows these titles or names positioned at the top of each column. A NIL pointer is passed as the column data format parameter, since this program only requires default display formats. Options are provided for saving and loading session and preference settings.

Figures 6.1a-b (p. 63) show two screens from the IGS used to create the graph in Figure 6.1c (p. 64). Many different types of plots are available using the same or a different data-set. This example attempts only to demonstrate the use of some of the services provided to programmers by the IGS. The next chapter more completely describes the user interface built into the IGS.

Table 7.2 - Data file for the IGS example program

79	25.34	22.33
80	36.4	13.29
81	42.33	56.3
82	23.49	39.1
83	32.1	35.94
84	89.3	74.78
85	12.3	10.23
86	49.4	32.65
87	78.87	87.78

Listing 7.1 - A complete IGS example program listing

```
Program Fig6_1;
Uses Crt,IGSType,IGSMain;  (* TP unit Crt plus two IGS units *)

Type R9x3 = Array[1..9,1..3] of REAL;  S9 = Array[1..9] of STRING[2];
Var Data : R9x3;  Year : S9; ColNames : ColTitleArrayPtr;  DatFile : TEXT;
    i,j : BYTE;  ColTypes : DataFieldTypeArrayPtr;  Load,Save : CHAR;
    Rows,Cols : INTEGER;  OK : BOOLEAN;

Begin
  LoadPrefs('Fig6_1.PRF');  (* Load the Preferences? section 7.2.6 *)
  OK:=False; New(ColNames); New(ColTypes);  (* Allocate pointer memory *)
  (* --- Set up the column headers (titles) --- *)
  ColNames^[1]:='Year'; ColNames^[2]:='Sample1'; ColNames^[3]:='Sample2';
  (* --- The data consists of three columns, the first being labels --- *)
  ColTypes^[1] := LabelField; For i := 2 to 3 do ColTypes^[i] := NumericField;

  (* --- Open the data file for reading --- *)
  Assign(DatFile,'Fig6_1.DAT'); Reset(DatFile);
  i := 0;
  While Not EOF(DatFile) do
    Begin
      Read(DatFile,Year[i+1]);  (* Read column 1, row i label field *)
      StorePointerInReal(@Year[i+1],Data[i+1,1]);  (* See section 7.2.7 *)
      For j := 2 to 3 do Read(DatFile,Data[i+1,j]);
      ReadLn(DatFile); i:=i+1;
    End;
  Rows:=i; Cols:=3;

  (* --- Load previously saved session? section 7.2.4 --- *)
  ClrScr; Write('Load the SESSION (Y or N) ? '); ReadLn(Load);
  If (UpCase(Load) = 'Y') And Not LoadSession('Fig6_1') Then
     WriteLn('Can''t load session.') Else OK:=True;

  If ( OK ) Then
     Begin
       (* --- Call the IGS, use default formatting, section 7.2.1 --- *)
       IGSMainMenu( @Data,Rows,Cols,ColTypes,NIL,ColNames );  TextMode(CO80);
       (* --- Save this session? --- *)
       Write( 'Save SESSION (Y or N) ? ' ); ReadLn( Save );
       If ( UpCase(Save) = 'Y' ) Then (* --- See sections 7.2.5,7.2.3 --- *)
          Begin ClrScr; SavePrefs('Fig6_1.PRF'); SaveSession( 'Fig6_1' ); End;
     End;

  (* --- Release the memory allocated for these pointers --- *)
  WriteLn; WriteLn('Bye.'); Dispose(ColTypes); Dispose(ColNames); IGSRelease;
End.
```

7.4. Dependencies

Access to seven IGS support routines is provided to TP programs by referencing the IGSMain unit. The programmer need only include the following TP unit reference statement to have the entire IGS available:

```
USES IGSType, IGSMain;
```

The IGS depends on two TP units (Crt and Graph) and the CGP and IGS units listed in Table 7.3. Additionally, the TP GU driver and stroked-font files must be available for use while the IGS is executing.

Table 7.3 - IGS and CGP support units required for compilation

Unit name	Description
CGP units	
CGPKPS	Kernel Plotting Services, see chapter two.
CGPHCS	Hard Copy Services, see chapter two.
CGPBPS	Basic Plotting Services, see chapter three.
CGP2DPS	Two-dimensional Plotting Services, see chapter four.
CGP3DPS	Three-dimensional Plotting Services, see chapter five.
IGS units	
IGSMain	Main IGS services: IGSMainMenu, IGSRelease, SaveSession, LoadSession, SavePrefs, LoadPrefs, StorePointerInReal.
IGSType	Public constant and type declarations.
IGSChrOp	String and character services.
IGSDsp	Text screen low level display services.
IGSEdScr	Screen editing services.
IGSErr	Error services, definitions, and constants.
IGSIntrp	Decodes plot screen input and drives the graphics routines.
IGSKbd	Keyboard services.
IGSRange	Range select and display services.
IGSWSCom	Worksheet constant, type declarations, and service routines.
IGSWSDsp	Worksheet display services.
IGSWSKbd	Worksheet keyboard services.
IGSWSPlt	Plot definition services.

Chapter 8 - IGS USER MANUAL

The IGS incorporates a user interface designed for ease-of-use and efficient operation. While using the IGS, various aspects of the user interface will be active. The main constructs in the user interface are:

Menus and Lists - Used to select one of several items.
Yes/No Prompts - Allow for verification of operations.
Edit Screens - Used to edit a full screen of information.

8.1. Menus and Lists

Menus and lists may appear as horizontal or vertical lists of choices or items. Initially an item is highlighted, and by using the arrow keys (←↑↓→), the cursor (highlight bar) may be moved to the next or previous item. When a desired item has been highlighted, pressing the <Enter> key will select that item. Also, for most menus and lists, pressing the first letter of a choice will select it directly. Generally, the <Esc> key is used to exit a menu or screen, but in some instances, a menu choice 'Exit' is used to exit.

8.2. Edit Screens

An edit screen consists of several fields positioned about the screen (see Figure 6.1b, p. 63). One field is highlighted at a time, and only the highlighted field may be edited. By using the arrow keys (←↑↓→) the cursor may be moved to an adjacent field. Also, the <Home> key will position the cursor in the first field, and the <End> key will position the cursor in the last field.

All plot specifications are made using edit screens customized according to the requirements of a particular plot type. Options (section 8.5.2) and preference (section 8.3.2) screens also use custom edit screens.

There are several different types of fields:

Text field - Used to enter alphanumeric information.
Numeric field - Used to enter numeric data.
Rotary field - Used to select one of a fixed set of choices.
Range field - Used to enter a range name.

8.2.1. Text/Numeric fields

To enter or edit a text or numeric field, type the desired information. If the field currently contains information, it is erased, and the new information is entered. Alternatively, if the field currently contains information, it can be edited using the keystrokes listed in Table 8.1 (p. 75).

Table 8.1 - Text/Numeric fields: Editing keys

Key	Description/Action
<Ctrl-Left-Arrow>	Move cursor one character to the left.
<Ctrl-Right-Arrow>	Move cursor one character to the right.
<Delete>	Delete character at cursor, pull characters left.
<BackSpace>	Delete previous character, move left, pulling characters.
<Insert>	Toggle between INSERT and OVERSTRIKE modes.

The cursor will flash as either a solid block or an underscore denoting the insert/overstrike mode. When a block cursor is flashing, the insert mode is active. All typed text will be inserted into the currently entered text. An underscore cursor means that typed text will replace (overstrike) existing text.

8.2.2. Rotary fields

Rotary fields contain single character choices or choice names. To select the next or previous choice, the <Ctrl-Right-Arrow> and <Ctrl-Left-Arrow> keys are used. For example, to select a font for use with the Main Header of a graph, a rotary field is used to cycle through the available fonts (see Graph Options screen, p. 82):

Main Header font: Default

selected from the rotary list:
Default
Triplex
Small
SansSerif
Gothic

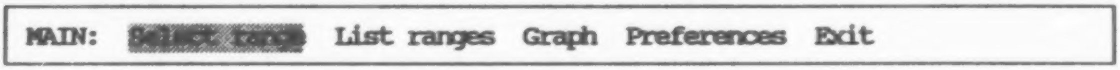
The current font appears and the other fonts are selected with the arrow keys (i.e., pressing <Ctrl-Right-Arrow> twice selects Small for the above example). Pressing <Enter> accepts the currently selected font.

8.2.3. Range fields

A range field (found only on plot definition screens), can be entered as a text field, or the <F1> function key may be pressed to allow the selection of a range name from a list of currently defined ranges. The <Esc> key will cancel such a selection and return to the edit screen. For more details, refer to section 8.5.1.

8.3. Main Menu

The IGS starts with a Main Menu and a spreadsheet containing the data array displayed on the screen (see Figure 6.1a, p. 63). The main Menu is used to select and list ranges, edit screen preferences, and select the Graph Menu. The menu is shown below; 'Select range' is initially highlighted:



MAIN: Select range List ranges Graph Preferences Exit

8.3.1. Ranges

A list of ranges may be defined by the user. Each range must have a unique name of up to 12 characters (the name is case sensitive; i.e., upper case letters are distinguished from lower case). A range is a single or group of contiguous blocks of data on the screen.

8.3.1.1. Select

To select a range, use the Main Menu 'Select range' option. A prompt for the range name will appear. Type a unique range name and press the <Enter> key. To cancel this operation, simply press the <Esc> key, or <Enter> key without typing a name. This will return control to the Main Menu.

Once the range name has been entered, the range may be specified. A highlighted cell pointer (CP) will appear on the spreadsheet. The CP may be moved freely to any data field in the spreadsheet by using any of the keys listed in Table 8.2 (p. 77). As the CP is moved, the current cell coordinates are displayed at the top of the screen.

A range consists of one or more range segments. Each segment consists of contiguous cells in one row or column. A segment is selected by toggling segment selection on in its first cell, and off in its last cell. Cell order within a segment may be the reverse of display order (i.e., may go from bottom to top in a column or right to left in a row).

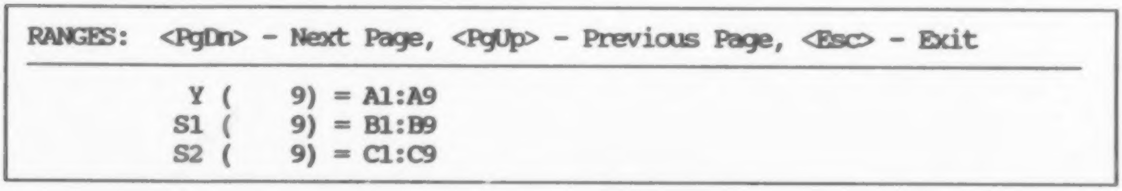
Table 8.2 - Range selection keys

Key	Description/Action
↑ <Up-Arrow>	Move up one row.
↓ <Dn-Arrow>	Move down one row.
← <Left-Arrow>	Move left one column.
→ <Right-Arrow>	Move right one column.
<Home>	Move to the first column (same row).
<End>	Move to the last column (same row).
<Ctrl-PgUp>	Move to the first row (same column).
<Ctrl-PgDn>	Move to the last row (same column).
<Enter>	Ends the current range selection, prompts for another range name.
<Esc>	Same as the <Enter> key.
. (dot)	Toggles range segment selection on and off.
<BackSpace>	Backs up and deletes/changes a range segment.

To edit a range that has been previously defined, the Main Menu item 'Select range' is used, and the name of the range to edit is entered. Then the <BackSpace> key can be used to remove range seqments and then new segments may be added.

8.3.1.2. List

To view all currently defined ranges, the Main Menu item 'List ranges' is used. This will result in a list of all ranges including range names, number of data fields, and range segment limits. This is a screen segment demonstrating a possible range list:



```
RANGES:  <PgDn> - Next Page, <PgUp> - Previous Page, <Esc> - Exit
-----------------------------------------------------------------
              Y (     9) = A1:A9
             S1 (     9) = B1:B9
             S2 (     9) = C1:C9
```

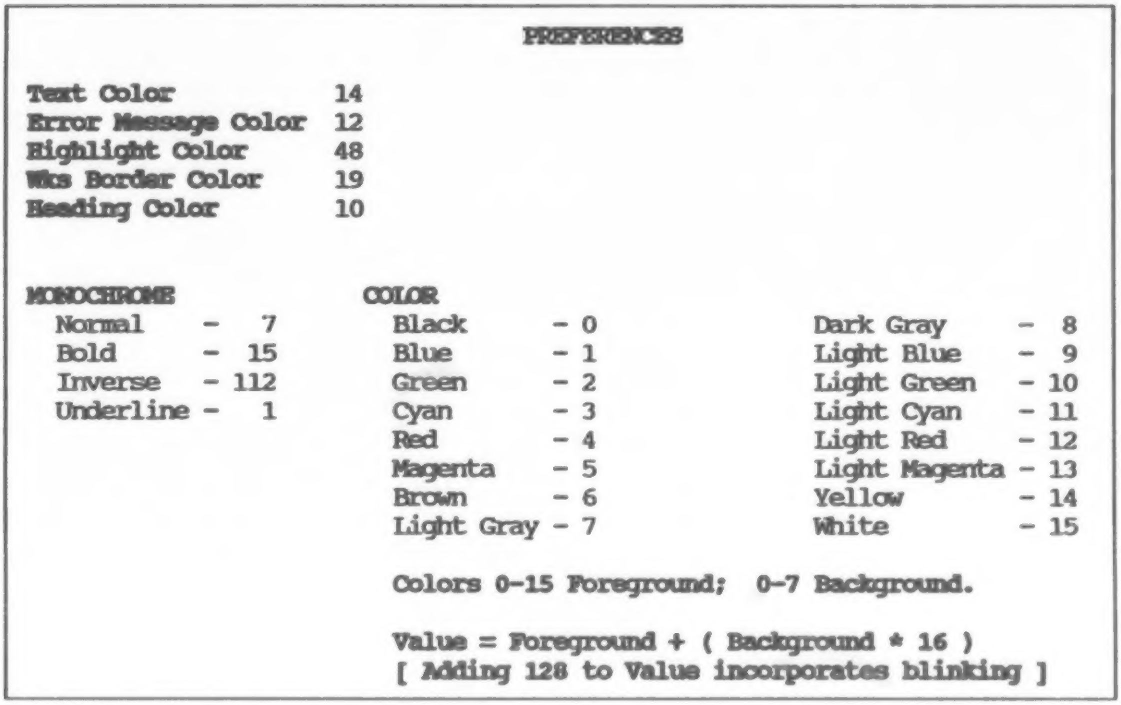
8.3.2. Preferences

By selecting the 'Preferences' option from the Main Menu, text screen attributes can be selected according to the user's display system and personal preferences. Display colors (or attributes for monochrome displays) may be changed for:

Text Color - includes all entered text, menu choices etc.
Error Message Color - error messages displayed by graphics system
Highlight Color - spreadsheet and menu highlight pointer
Wks Border Color - worksheet border: A B C... and 1 2 3...
Heading Color - screen titles, prompts, etc.

To establish a color and attribute, a number from 0 to 255 must be entered in the appropriate field. The effect of the number depends on the type of display adapter and monitor used. Figure 8.1 (p. 79) shows color selections for the five display categories. They translate to:

Text Color 14 → Yellow
Error Message Color 12 → LightRed
Highlight Color 48 → Black on Cyan
Wks Border Color 19 → Cyan on Blue
Heading Color 10 → Light Green



```
                              PREFERENCES

Text Color             14
Error Message Color    12
Highlight Color        48
Wks Border Color       19
Heading Color          10


MONOCHROME             COLOR
  Normal    -   7        Black       - 0             Dark Gray     -  8
  Bold      -  15        Blue        - 1             Light Blue    -  9
  Inverse   - 112        Green       - 2             Light Green   - 10
  Underline -   1        Cyan        - 3             Light Cyan    - 11
                         Red         - 4             Light Red     - 12
                         Magenta     - 5             Light Magenta - 13
                         Brown       - 6             Yellow        - 14
                         Light Gray  - 7             White         - 15

                         Colors 0-15 Foreground;  0-7 Background.

                         Value = Foreground + ( Background * 16 )
                         [ Adding 128 to Value incorporates blinking ]
```

Figure 8.1 - IGS Preferences screen

Monochrome (Hercules) Adapter attribute values are given in Figure 8.1 under the bolded heading **MONOCHROME**. Color attribute values consist of a foreground and background color. To determine the value for a given foreground and background color, use the values given in Figure 8.1. To arrive at a color attribute value use this formula:

ColorValue = ForegroundColor + (BackgroundColor * 16)

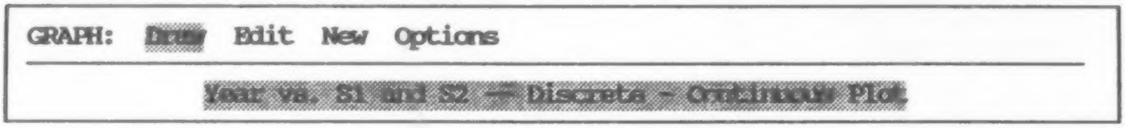
Adding 128 to the attribute value makes the foreground blink. Also note that only colors 0-7 can be used for background color.

8.3.3. Exit (saving)

To exit or quit from the IGS, the IGS Main Menu item 'Exit' is used. Selecting Exit causes the IGS to return program control to the programmer's routine that initiated the call to the IGS. The IGS does not automatically save the changes made during execution. The IGS services **SaveSession** and **SavePrefs** must be called by the programmer's program to save the state of the IGS in disk files which may be reloaded in a later session. (See sections 7.2.3 - 7.2.6).

8.4. Graph Menu

From the Graph Menu, graph related options are selected. Plot definitions are used to describe how graphs are to be drawn. These definitions can be entered and edited, and then a graph may be plotted according to the definition. There are four options available from this menu:

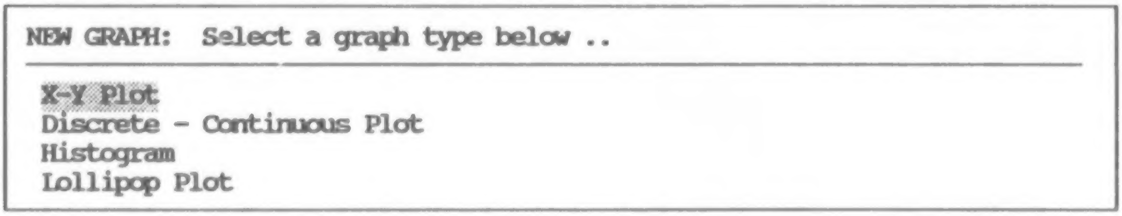


GRAPH: Draw Edit New Options

Year vs. S1 and S2 -- Discrete - Continuous Plot

All currently defined plots are listed below the horizontal line. The first plot definition is highlighted. By moving the cell pointer up and down using the arrow keys (↑↓), a different plot definition may be selected. The two options 'Draw' and 'Edit' act on the currently selected plot definition.

To create a new plot definition, the 'New' option is selected from the Graph Menu. A list of the four plot definition types (see below) is displayed and one may be selected, or the <Esc> key may be pressed to return to the Graph Menu. Initially, the first type is highlighted.



NEW GRAPH: Select a graph type below ..

X-Y Plot
Discrete - Continuous Plot
Histogram
Lollipop Plot

After selecting a plot definition type, a prompt requesting a name for the plot definition appears. After entering a name for the plot definition, a plot definition screen appears, specifically designed for the particular plot type. At this point the screen may be filled in.

Once a plot is defined, it may be drawn by highlighting the named plot and selecting the 'Draw' option. Pressing any key while viewing the graph returns control to the Graph Menu. If any errors are detected by the system a list of messages will appear; pressing any key returns control to the Graph Menu.

8.5. Plot specifications

By choosing 'New' from the Graph Menu or highlighting an existing plot definition and choosing 'Edit', a plot definition may be specified or changed. This section describes in detail each plot type definition screen, including how to enter information into a screen.

8.5.1. Named ranges as plot variables

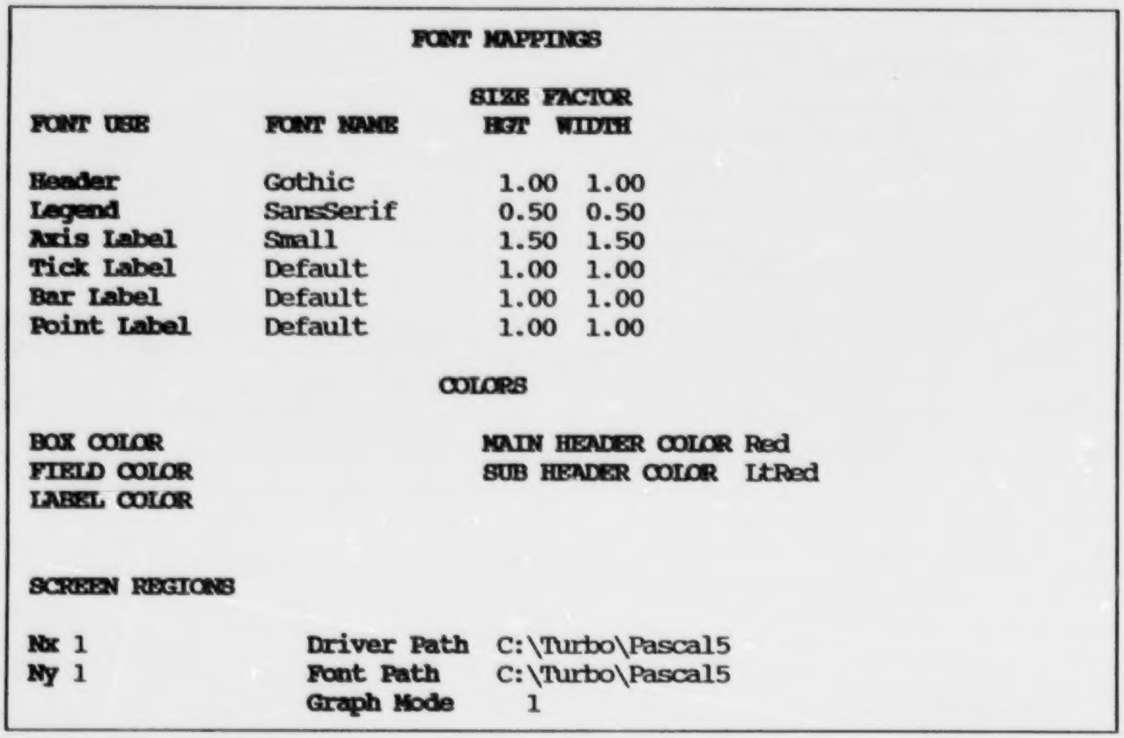
Named ranges defined from the spreadsheet are used as variables in plotting. They are entered in appropriate places in plot specification screens, as described below. These ranges may contain real numbers, symbols, or both.

When specifying graphs, it is important to respect the distinction between numbers and symbols. Some plot variables are required to be continuous, and the ranges used for them must contain only numbers. Other variables are discrete. The IGS allows ranges containing symbols or numbers or a mixture to be used as discrete variables. When real numbers appear in ranges specified as discrete plotting variables, they are converted to strings in the same format as in the spreadsheet display. The programmer may control this format, as described in chapter seven.

Note that symbols (character strings) may contain digits, and while viewing the spreadsheet it is possible to mistake a column of symbols made up only of digits and decimal points as a column of numbers. In this case, it is the user's knowledge of the data supplied to the IGS which will enable the distinction to be made.

8.5.2. Options screen

Some common aspects of plot generation are controlled by variables set on the Options screen (Figure 8.2).



```
                          FONT MAPPINGS

                                       SIZE FACTOR
FONT USE          FONT NAME            HGT  WIDTH

Header            Gothic               1.00  1.00
Legend            SansSerif            0.50  0.50
Axis Label        Small                1.50  1.50
Tick Label        Default              1.00  1.00
Bar Label         Default              1.00  1.00
Point Label       Default              1.00  1.00

                              COLORS

BOX COLOR                              MAIN HEADER COLOR Red
FIELD COLOR                            SUB HEADER COLOR  LtRed
LABEL COLOR


SCREEN REGIONS

Nx 1                 Driver Path   C:\Turbo\Pascal5
Ny 1                 Font Path     C:\Turbo\Pascal5
                     Graph Mode      1
```

Figure 8.2 - Graph Menu: Options screen

Font styles for headers and labels are selected by rotary fields from among the five available in TP (see Table 3.3, p. 34). Size is determined by two scaling factors, one applied to character width, the other to character height. The default size depends on the font selected. Size factors may be less than or greater than 1.0. The Default font behaves differently from the others in that it scales equally in both directions and may only be scaled up.

The IGS makes no attempt to automatically scale text to fit into the space available. It is up to the user to ensure that fonts and scaling factors are appropriate. Default font sizes are expressed in pixels, so that a given amount of text in a given font and size will take up less room on a screen with higher resolution.

Several color options may be selected by rotary fields. Note that a full range of 16 color names is available in the color rotary fields, but what will appear on the screen depends upon the graphics device and mode. If a color is chosen which is not available, the default drawing color will be used.

There is provision for specifying directory paths for the driver and font files. If these files are not in the current directory these paths must be given.

The IGS provides a facility for drawing more than one graph on the screen. The screen may be divided in half horizontally, vertically, or both, thus yielding one, two, or four drawing areas. This is controlled by the setting of the **Nx** and **Ny** rotary fields. Note that the use of multiple screen divisions results in extra memory allocation for image storage. On systems with EGA or VGA devices this can be a considerable amount of memory.

Finally, graph mode is selected on the Options screen. The modes available depend upon the graphics device in use. Table 8.3 (p. 83) is provided to assist with mode selection. See the TP Reference Manual (Borland 1988b) for more details.

Table 8.3 - IGS Graphics Options: Graph Mode options

Graphics driver	Mode	Resolution	Colors
CGA	0	320 x 200	LightGreen,LightRed,Yellow
(Includes MCGA	1	320 x 200	LightCyan,LightMagenta,White
and ATT400)	2	320 x 200	Green,Red,Brown
	3	320 x 200	Cyan, Magenta, LightGray
	4	640 x 200	Monochrome
MCGA	5	640 x 350	Monochrome
EGA	0	640 x 200	16 color
	1	640 x 350	16 color
EGA64	0	640 x 200	16 color
	1	640 x 350	4 color
VGA	0	640 x 200	16 color
	1	640 x 350	16 color
	2	640 x 480	16 color
Hercules Mono	0	720 x 348	Monochrome

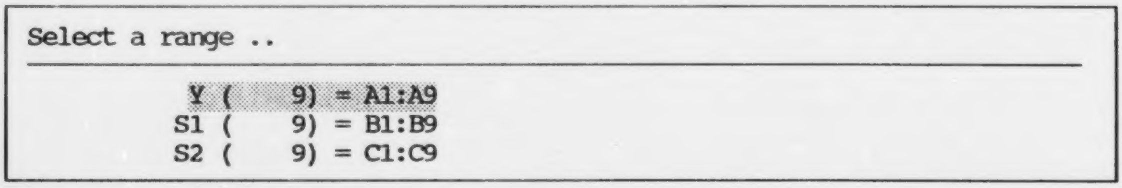
8.5.3. Plot definition screens

There is a plot definition screen for each of the four plot types supported by the IGS, including one for an X-Y Plot, Discrete-Continuous Plot, Histogram, and Lollipop Plot. Some material is common to all screens. When a new plot definition is initiated, the user is prompted for a plot name. This name will appear on the plot definition screen whenever it is displayed. At the bottom of each screen there is a section in which the text to be drawn as headings and axis labels is specified. Each of these is a text field. To the right of this section is a collection of Y/N (Yes/No) rotary fields which control plot layout features. The **GRID** field controls the drawing of a grid at major tick marks for each axis. For tick marks (i.e., major tick marks) to be drawn a "Y" must be entered in the **AXES** field. A "Y" in the **MINOR TICKS** field causes minor tick marks to be drawn. To have a box or outline drawn around the plot, enter a "Y" in the VP_BOX field. The **BOT**, **TOP**, **LFT**, and **RGT** fields control the drawing of the bottom, top, left, and right axis lines.

Depending on the type of plot, the screen has fields for specifying one, two, or three continuous axes. The **TYPE** field is a rotary by which a linear (Lin), logarithmic (Log) or probability (Prb) axis may be selected. The appropriate transformation will be performed automatically. If the axis extrema are out of range for Log or Prb, a linear axis will be used. The **MODE** field (scaling) is a rotary, taking the values A, R, or M. The value A (Automatic) specifies that the extrema for the axis are to be derived from the data. R (Round) indicates that the extrema are to be taken from the data and then rounded to *nice* numbers. If M (Manual) is selected, the user may enter explicit extrema in the MIN and MAX fields, which are number editing fields. If a field is left blank, the extremum will be taken from the data.

All screens contain a SYMBOL SIZE field. This is a number editing field in which the user may enter a factor by which the symbol size will be scaled.

All screens provide for the plotting of ten sets of data, of which any number may be used. The user must enter the names of ranges which will be used as discrete or continuous variables. This may be done by typing in the name or by selecting from a menu which is brought up by pressing the <F1> key while the cursor is on the X, Y, or Z field, depending on the plot type (see Figure 8.3).



```
Select a range ..

          Y (    9) = A1:A9
         S1 (    9) = B1:B9
         S2 (    9) = C1:C9
```

Figure 8.3 - Range name selection screen

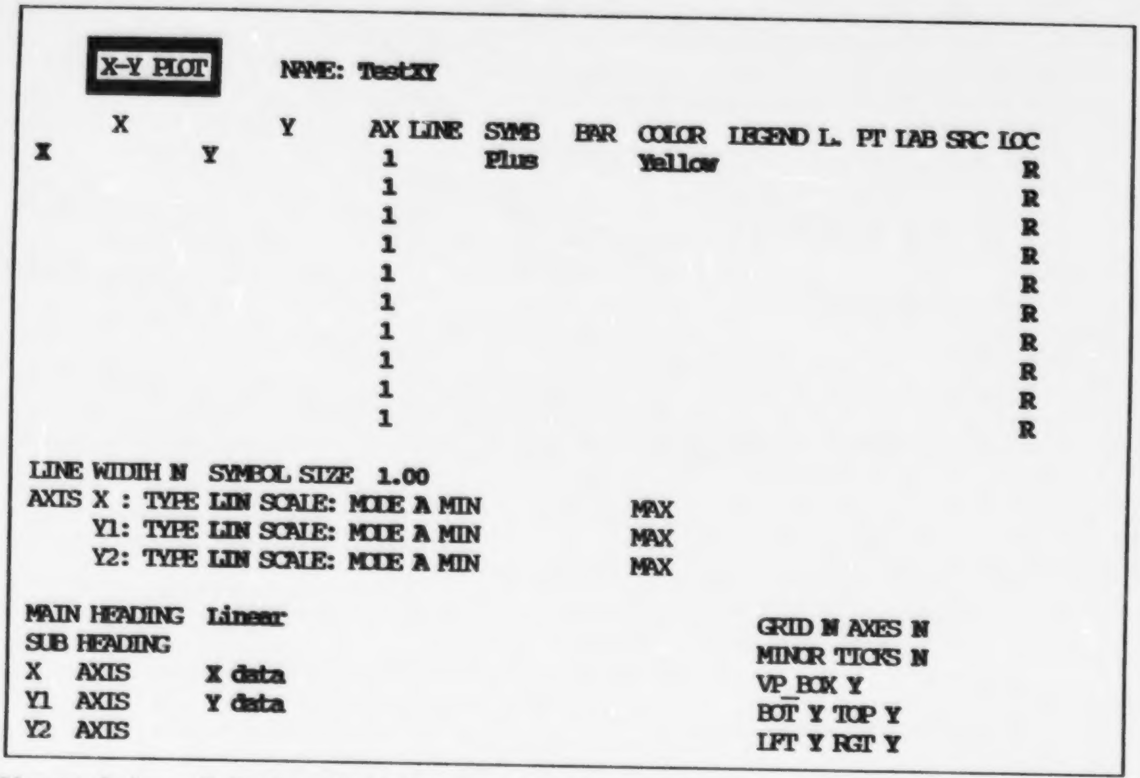
When the list of range names appears, the first name is highlighted. The <Up-Arrow> and <Dn-Arrow> keys (↑↓) may be used to move the cursor to the desired range name. Pressing <Enter> selects the currently highlighted name and enters it on the plot specification screen, moving the cursor to the next field. While in the range name selection screen, the user may return to the plot screen without making a selection by pressing the <Esc> key.

Some screens provide for point labelling. In that case, the **LOC** field is used to specify the placement of the text string relative to plotted points. This is a rotary field with the possible values R (string to right of point), L (to left), T (above the point), B (below), or C (centered on the point).

When multiple variables are drawn on one graph they may be labelled by means of a legend at the bottom of the graph. If the **LEGEND** L. (legend label) field is filled in (this is a text field), the entered text will be used to label the legend entry.

8.5.3.1. X-Y plot

An X-Y plot is configured using the screens shown in Figures 8.4a-b (pp. 86-87). In specifying an X-Y plot, the user must enter range names to be used as X and Y variables. The first X field must be filled. Subsequent X fields may be left blank, in which case the X variable will be taken from the next filled field above the blank one. If any Y field is blank, even if an X variable is specified, no line or symbol will be drawn for the X,Y pair.



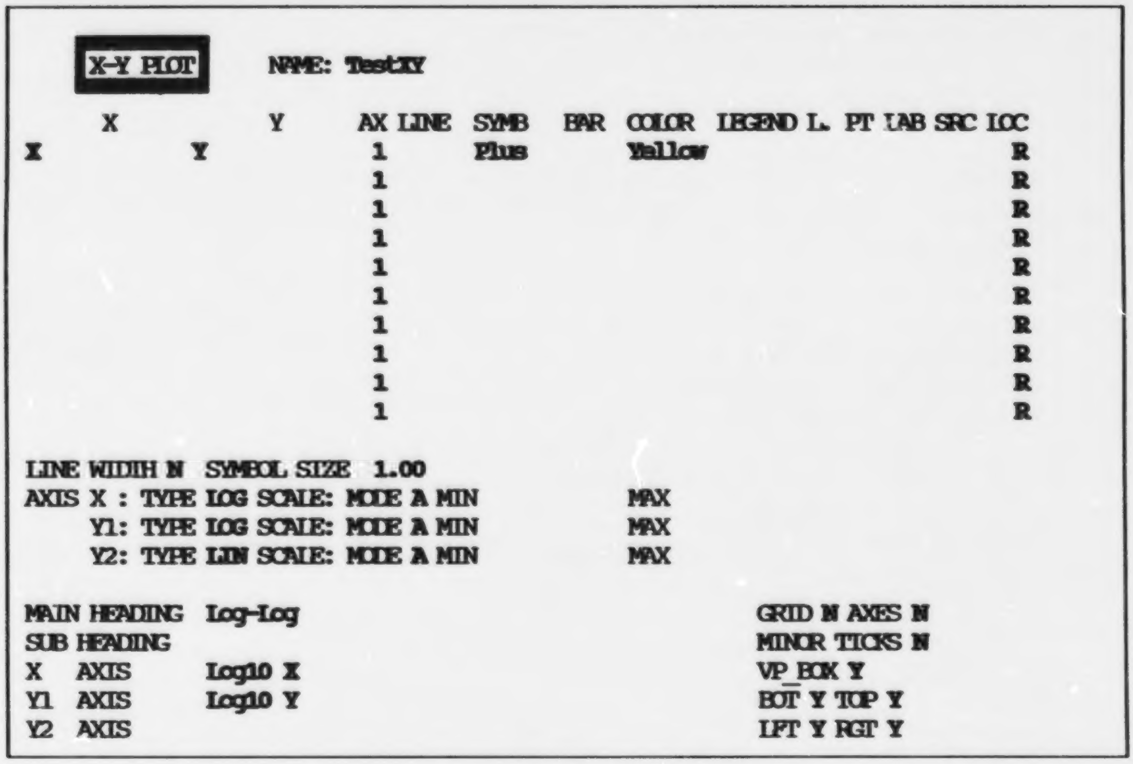
```
    X-Y PLOT        NAME: TestXY

         X              Y       AX LINE  SYMB    BAR  COLOR  LEGEND L. PT LAB SRC LOC
X              Y                 1       Plus         Yellow                        R
                                 1                                                  R
                                 1                                                  R
                                 1                                                  R
                                 1                                                  R
                                 1                                                  R
                                 1                                                  R
                                 1                                                  R
                                 1                                                  R
                                 1                                                  R
LINE WIDTH N  SYMBOL SIZE  1.00
AXIS X : TYPE LIN SCALE: MODE A MIN             MAX
     Y1: TYPE LIN SCALE: MODE A MIN             MAX
     Y2: TYPE LIN SCALE: MODE A MIN             MAX

MAIN HEADING  Linear
SUB HEADING                                               GRID N AXES N
X   AXIS      X data                                      MINOR TICKS N
Y1  AXIS      Y data                                      VP_BOX Y
Y2  AXIS                                                  BOT Y TOP Y
                                                          LFT Y RGT Y
```

Figure 8.4a - X-Y plot definition screen example part 1. Produces the first half of Figure 8.5 (p. 88).

By setting the Graph Options screen **Nx** field to 2 the screen is divided into two independent regions. The plot screen shown in Figure 8.4a generates the graph in the left half of Figure 8.5 (p. 88). Similarly, the plot screen shown in Figure 8.4b (p. 87) generates the graph in the right half of Figure 8.5.



```
     X-Y PLOT      NAME: TestXY

       X             Y         AX LINE  SYMB   BAR  COLOR  LEGEND L. PT LAB SRC LOC
X            Y                 1        Plus        Yellow                       R
                               1                                                 R
                               1                                                 R
                               1                                                 R
                               1                                                 R
                               1                                                 R
                               1                                                 R
                               1                                                 R
                               1                                                 R
                               1                                                 R

LINE WIDTH N  SYMBOL SIZE  1.00
AXIS X : TYPE LOG SCALE: MODE A MIN            MAX
     Y1: TYPE LOG SCALE: MODE A MIN            MAX
     Y2: TYPE LIN SCALE: MODE A MIN            MAX

MAIN HEADING  Log-Log                                        GRID N AXES N
SUB HEADING                                                  MINOR TICKS N
X   AXIS      Log10 X                                        VP_BOX Y
Y1  AXIS      Log10 Y                                        BOT Y TOP Y
Y2  AXIS                                                     LFT Y RGT Y
```

Figure 8.4b - X-Y plot definition screen example part 2. Produces the second half of Figure 8.5 (p. 88).

An X-Y plot may have one or two dependent variable (Y) axes. If one is used, it is drawn on the left side of the plot. If two are specified, the first is on the left side and the second on the right. Which axis to use is determined for an X,Y pair by the value of the **AX** field, which is a rotary field with values 1 and 2. These axes are called Y1 and Y2, respectively in the axis specification section of the screen.

Each X,Y variable set may be represented by lines, symbols, or bars, or any combination of these. The **LINE**, **SYMBOL**, and **BAR** fields are rotaries which allow the selection of nothing (blank) or one of the possibilities for the corresponding field (line style for **LINE**, symbol type for **SYMBOL** (Table 3.4, p. 38), and fill style for **BAR** (Table 4.2, p. 46)). The **COLOR** field, which is a rotary for the 16 possible colors (depends on hardware), determines the color of whatever data representation is chosen.

When the **BAR** option is chosen, the data points are represented as vertical bars centered on the data points. In the case of multiple X,Y pairs, the bars are grouped side-by-side. If bars are to be chosen, the X axis extrema must be specified to be sufficiently beyond the extreme data points to allow space for the bars to be drawn.

Individual points on a graph may be labelled with character strings from a named range specified in the **PT LAB SRC** field. This field may be filled from the menu of named ranges (i.e., using the <F1> key).

Figure 8.5 illustrates a typical X-Y plot. The X data varies from 1 to 25 and the Y data from 1^2 to 25^2. To create two regions, the **Nx** field on the Graph Options screen is set to 2. Also, the Header font is set to **SansSerif**. The plot shown in the second region is set up by setting the X and Y1 **TYPE** fields to LOG (Logarithmic) and the **SCALE: MODE** fields set to R (Round).

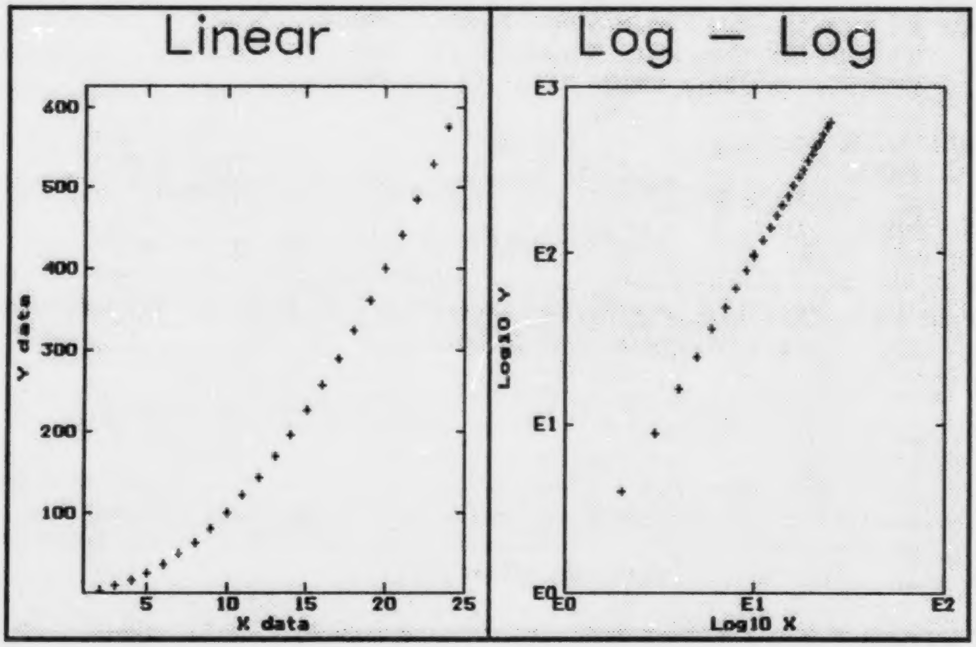
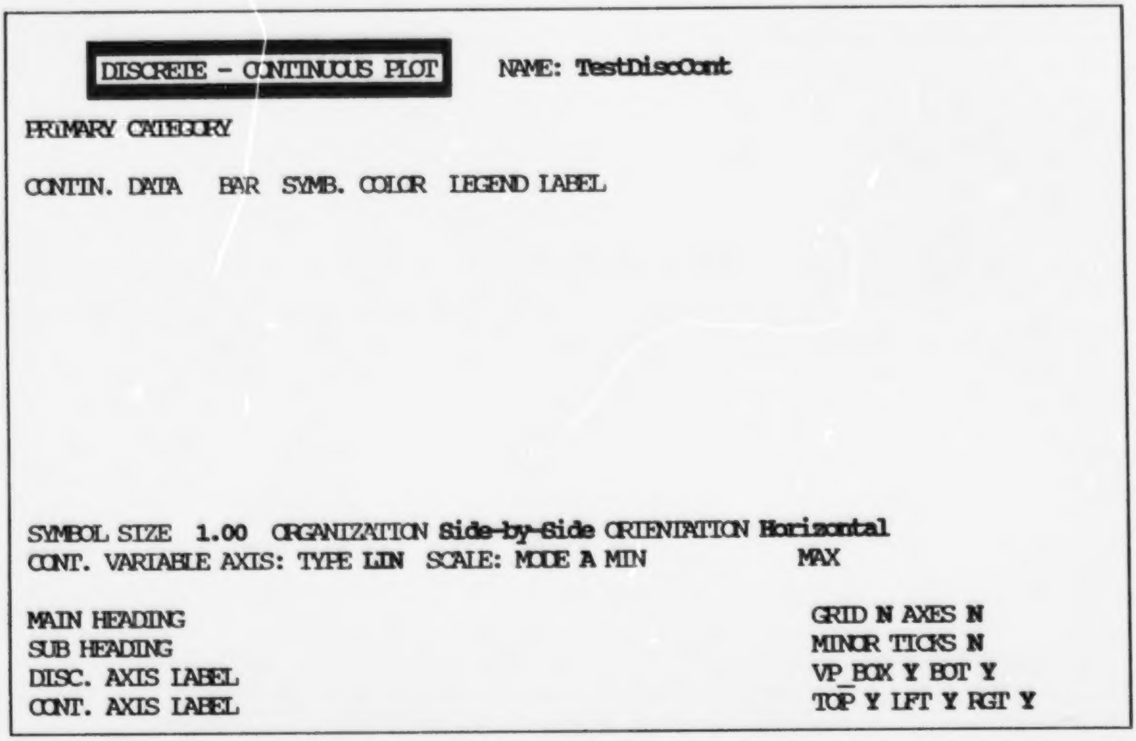


Figure 8.5 - Example of an IGS X-Y plot

8.5.3.2. Discrete-Continuous plot

In specifying a bar chart, the user must enter a range name to be used as the independent discrete-valued variable which specifies primary categories (bars or groups of bars). Then up to 10 continuous variables may be selected, each of which will generate a set of bars for each primary category. Figure 8.6 (p. 89) shows the definition screen the user must fill in to set up a Discrete-Continuous plot.



```
   DISCRETE - CONTINUOUS PLOT     NAME: TestDiscCont

PRIMARY CATEGORY

CONTIN. DATA   BAR  SYMB. COLOR  LEGEND LABEL








SYMBOL SIZE  1.00  ORGANIZATION Side-by-Side ORIENTATION Horizontal
CONT. VARIABLE AXIS: TYPE LIN  SCALE: MODE A MIN          MAX

MAIN HEADING                                            GRID N AXES N
SUB HEADING                                             MINOR TICKS N
DISC. AXIS LABEL                                        VP_BOX Y BOT Y
CONT. AXIS LABEL                                        TOP Y LFT Y RGT Y
```

Figure 8.6 - Discrete-Continuous plot definition screen

The **BAR** field is a rotary in which bar fill style is selected. Symbols (of which the type is chosen in the rotary field **SYMB**) may be used as well as or instead of bars to represent the data. Bar and symbol color is selected by the rotary field **COLOR**.

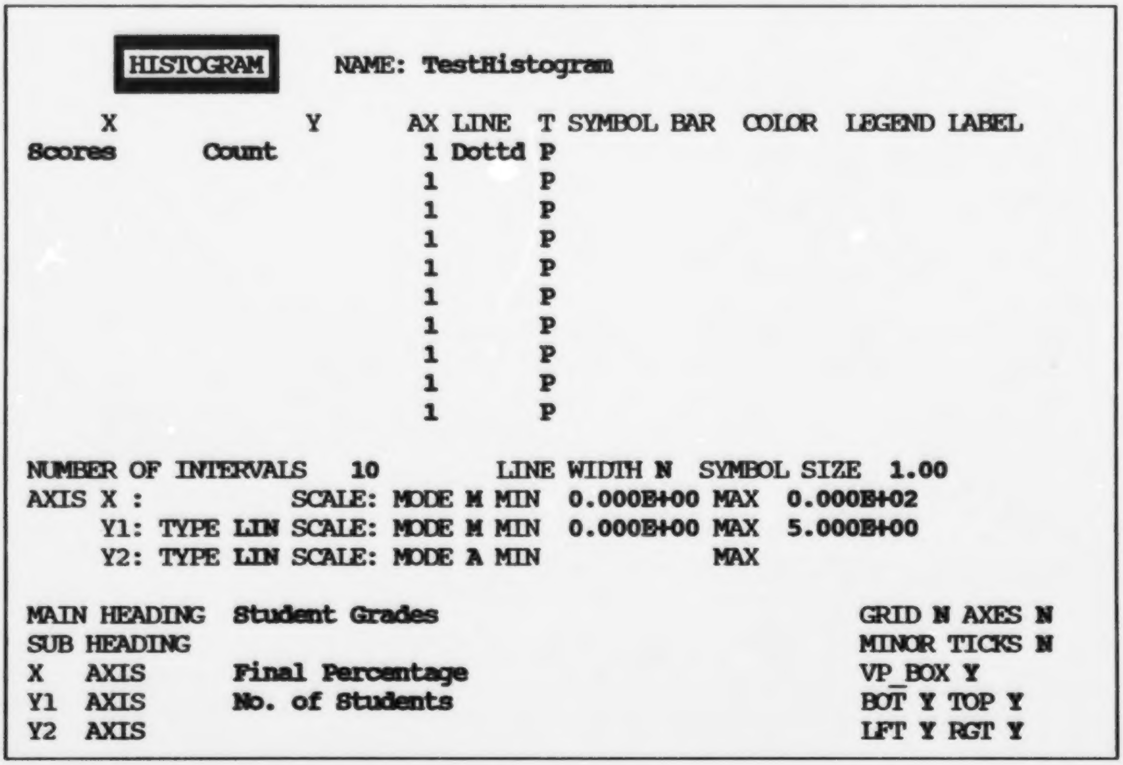
Bar charts with multiple dependent variables may be organized with the bars side-by-side, stacked, or stacked and scaled to 100%, according to the value in the rotary field **ORGANIZATION**.

The field **ORIENTATION** is a rotary having the values **Horizontal** and **Vertical**. If the former is chosen, the independent variable axis is up the left side of the plot and bars are drawn horizontally. This organization gives more room for drawing the primary category labels.

See Figures 6.1a-c (pp. 63-64) for an example of a Discrete-Continuous plot.

8.5.3.3. Histogram plot

The field **NUMBER OF INTERVALS**, which is to be filled with an integer in the range 1..999, specifies the number of equal intervals into which the X axis is to be divided. Note that the X axis may only be linear. See Figure 8.7 for a picture of the initial definition screen to be filled out for a Histogram plot.



```
    HISTOGRAM       NAME: TestHistogram

    X            Y      AX LINE  T SYMBOL BAR  COLOR  LEGEND LABEL
Scores      Count        1 Dottd P
                         1       P
                         1       P
                         1       P
                         1       P
                         1       P
                         1       P
                         1       P
                         1       P
                         1       P

NUMBER OF INTERVALS   10       LINE WIDTH N  SYMBOL SIZE  1.00
AXIS X :           SCALE: MODE M MIN  0.000E+00 MAX  0.000E+02
     Y1: TYPE LIN SCALE: MODE M MIN  0.000E+00 MAX  5.000E+00
     Y2: TYPE LIN SCALE: MODE A MIN             MAX

MAIN HEADING  Student Grades                          GRID N AXES N
SUB HEADING                                           MINOR TICKS N
X   AXIS      Final Percentage                        VP_BOX Y
Y1  AXIS      No. of Students                         BOT Y TOP Y
Y2  AXIS                                              LFT Y RGT Y
```

Figure 8.7 - Histogram plot definition screen

For each X,Y pair, the following interpretation holds: if a range name is entered for X, then the system accumulates the X,Y array element pairs into the appropriate X axis *bin*. The ranges may specify more elements than there are bins. If a bin is not specified by an element pair, the value in that bin will be zero. If no X range is specified, the system expects the Y range to contain one element for each X axis interval, and the value of each element is taken directly as the histogram value. If no X ranges are specified, the X axis **MODE** must be **M** and the extrema must be explicitly specified.

Figure 8.8 (p. 91) illustrates a typical Histogram plot. The X data are student scores and the Y data are numbers of students per score. The Header font is set to **SansSerif**. The X and Y1 **SCALE: MODE** fields are set to **M** (manual) with X ranging from 0.0 to 100.0 and Y1 ranging from 0.0 to 5.0. **V** (vertical) is the selected histogram type. The **NUMBER OF INTERVALS** field is set to 10.

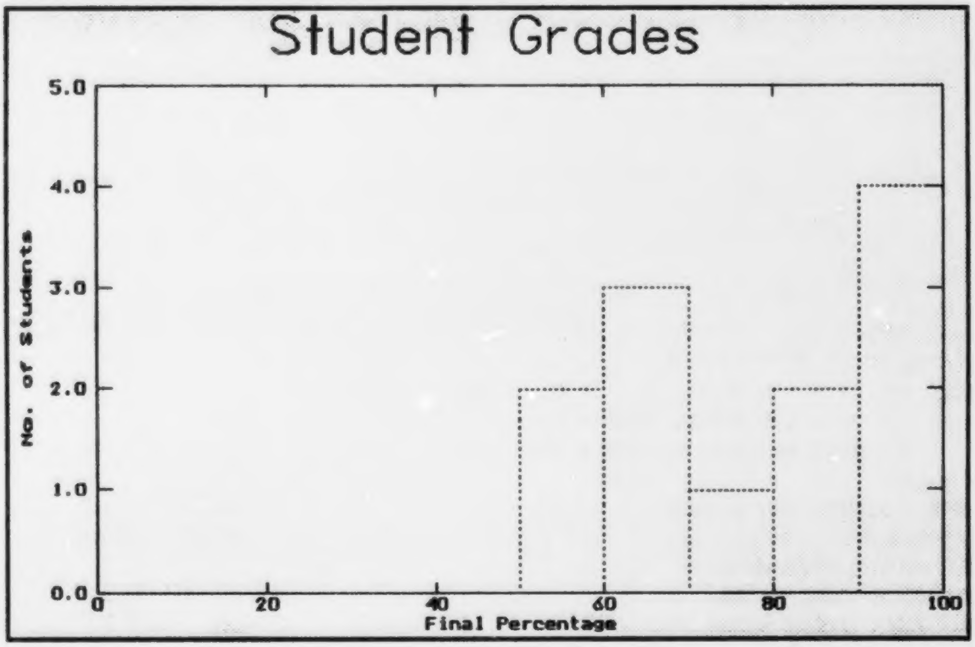
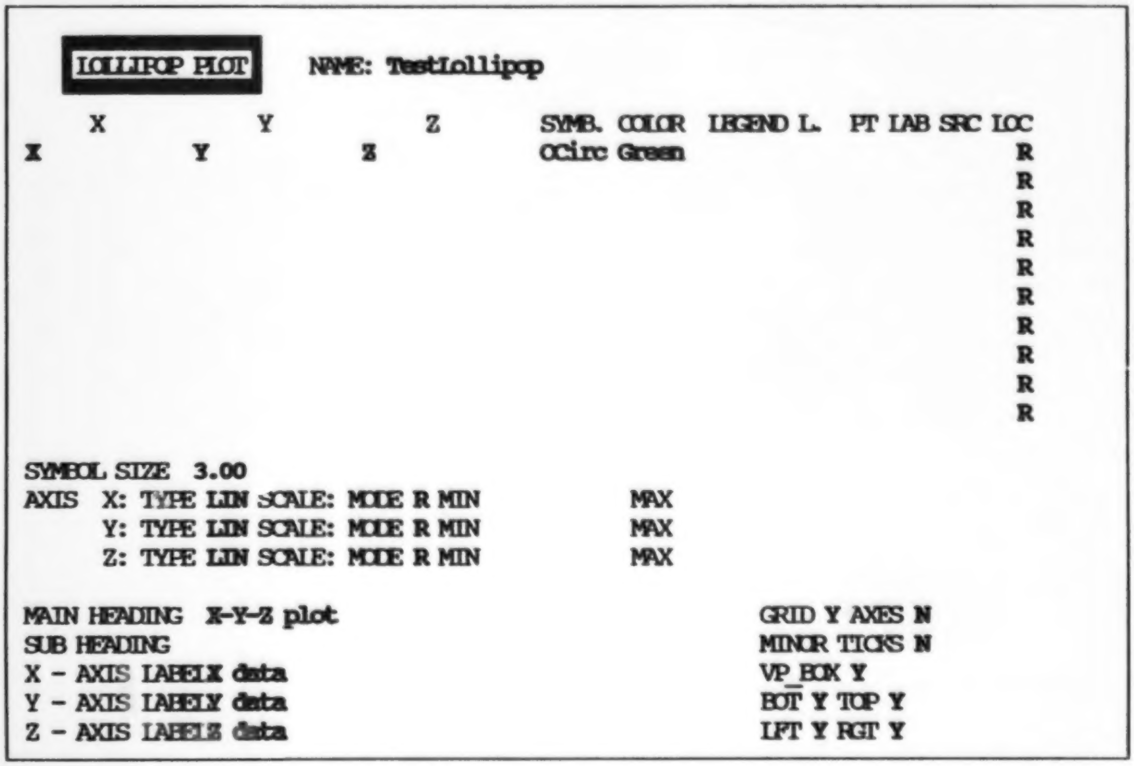


Figure 8.8 - Example of an IGS Histogram plot

8.5.3.4. Lollipop plot

Lollipop plots are 3D scatter plots represented in perspective. To generate such a plot, X,Y,Z triplets of range names must be specified. Each data point will be represented by a symbol of type selected by the rotary field **SYMB**. The **COLOR**, **LEGEND L.**, **PT LAB SRC**, and **LOC** fields have the same meanings as in other screens. See the associated definition screen in Figure 8.9.



```
   LOLLIPOP PLOT     NAME: TestLollipop

     X               Y              Z         SYMB. COLOR  LEGEND L.  PT LAB SRC LOC
X              Y              Z               OCirc Green                          R
                                                                                   R
                                                                                   R
                                                                                   R
                                                                                   R
                                                                                   R
                                                                                   R
                                                                                   R
                                                                                   R
                                                                                   R

SYMBOL SIZE  3.00
AXIS  X: TYPE LIN SCALE: MODE R MIN          MAX
      Y: TYPE LIN SCALE: MODE R MIN          MAX
      Z: TYPE LIN SCALE: MODE R MIN          MAX

MAIN HEADING  X-Y-Z plot                                  GRID Y AXES N
SUB HEADING                                               MINOR TICKS N
X - AXIS LABELX data                                      VP_BOX Y
Y - AXIS LABELY data                                      BOT Y TOP Y
Z - AXIS LABELZ data                                      LFT Y RGT Y
```

Figure 8.9 - Lollipop plot definition screen

Figure 8.10 illustrates a typical Lollipop plot. The X, Y, and Z data are a set of (x,y,z) triplets. The Header font is set to SansSerif. The X, Y, and Z **SCALE: MODE** fields are set to R (round). The **SYMBOL SIZE** field is set to 3.0.

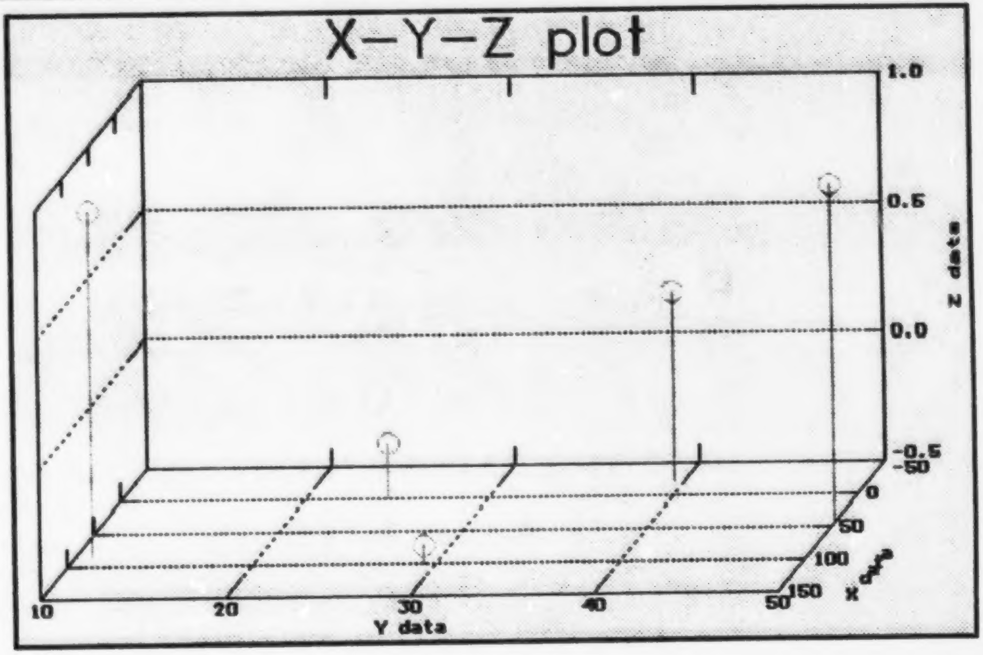


Figure 8.10 - Example of an IGS Lollipop plot

PART THREE

CUSTOM
PLOTTING
PACKAGE
(CPP)

Chapter 9 - CPP INTRODUCTION
Chapter 10 - CPP USER MANUAL
Chapter 11 - CPP DATA FILE FORMAT

Chapter 9 - CPP INTRODUCTION

The Custom Plotting Package (CPP) is a menu-driven system for loading data files and generating graphics on microcomputer display screens. It uses the Interactive Graphics System (IGS) described in Part II of this document. Users should refer to chapter eight for descriptions of all IGS related functions.

Users of the CPP need not have computer programming knowledge or skills. The CPP is a stand-alone computer program. Users create a data file specific to their particular requirements and then named ranges and plot definitions are defined via the IGS. All named ranges and plot definitions can be saved for use in future sessions of the CPP.

This document provides a user's manual (chapter 10) which describes how to load data files, how to load and save CPP sessions, and how to pass control to the IGS. Chapter 11 discusses the data file format supported by the CPP file loader. Graphics printing is currently not supported from the CPP menus, however, the PrintImage program (see part V) supports draft quality image printing.

Chapter 10 - CPP USER MANUAL

The CPP is a simple front-end data manager for the IGS. From the CPP menu, users are provided with a few commands for loading data files, loading and saving CPP sessions, and moving into the IGS to produce graphs.

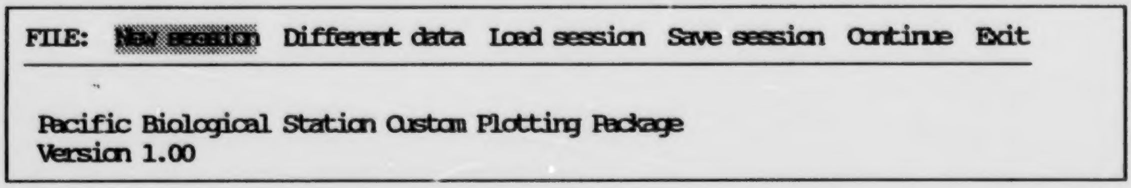
10.1. Overview

When the CPP is invoked the user must instruct the program to load a data file or a previously saved CPP session. If the file(s) are successfully loaded the CPP passes control to the IGS. The user may then select data ranges from the data and produce various kinds of plots.

Different data-sets may be loaded during the same session. This feature may be used to produce a particular set of plots for different data-sets with the same format.

10.2. Commands

The CPP begins with a File Menu which provides commands for loading data and saving or loading sessions. The menu is shown in Figure 10.1; 'New session' is initially highlighted:



FILE: New session Different data Load session Save session Continue Exit

Pacific Biological Station Custom Plotting Package
Version 1.00

Figure 10.1 - Custom Plotting Package FILE menu

10.2.1. New session

To begin a new session with the CPP, choose the 'New session' command. If a previous session was active, all previously named ranges and plot definitions are erased from memory. Sessions required for future use should be saved prior to beginning a new session. When this command is chosen, the user is required to respond to the following prompts:

Starting a new session clears memory .. Continue (Y/N)?

[Answer 'N' to abandon this action or 'Y' to proceed (see below).]

Enter data filespec:

[Enter the full name of the data file to load (including path and extension). Pressing the <ESC> or <Enter> key without an entry, cancels this operation.]

When a data file is specified, it is loaded, and control is then passed to the IGS. At this point all IGS functions are available. Exiting from the IGS returns to the File Menu of the CPP.

10.2.2. Different Data

Named ranges and plot definitions established in the IGS may be applied to different data-sets. Selecting the 'Different data' command allows for a different data-set to be loaded while maintaining the currently defined named ranges and plot definitions. When this command is chosen, the user must respond to the following prompt:

Enter data filespec:

[Enter the full name of the data file to load (including path and extension). Pressing the <ESC> or <Enter> key without an entry, cancels this operation.]

When a data file is specified, currently named ranges and plot definitions are temporarily written out to three disk files (Plot.X00, Plot.X01, and Plot.X02) and the data file is then loaded.

10.2.3. Save session

Named ranges and plot definitions may be saved for future use by selecting the 'Save session' command. Choosing this command requires the user to enter a session filename:

Enter session filename to save:

[Enter the name of the session files to save (including path; excluding extension). Pressing the <ESC> or <Enter> key without an entry, cancels this operation.]

When a session file is specified, currently named ranges and plot definitions are written out to three disk files, using the specified filename and the extensions X00, X01, and X02.

10.2.4. Load session

Named ranges and plot definitions saved in a previous session may be loaded with the 'Load session' command. Choosing this command requires the user to respond to the following two prompts:

Enter session filename to load:

[Enter the name of the session files to load (including path; excluding extension); see the next prompt. Pressing the <ESC> or <Enter> key without an entry, cancels this operation.]

Enter data filespec:

[The name of the data file that was in use when the session was saved is offered. Pressing <Enter> accepts the filename (it may be changed). Press the <ESC> key to abandon this operation.]

When a session file is specified, all currently defined ranges and plot definitions are cleared, and the saved ranges, plot definitions, and data are loaded. Also, control is passed to the IGS.

10.2.5. Continue

The 'Continue' command allows reentry to the IGS with the currently defined named ranges, plot definitions, and loaded data.

10.2.6. Exit

Invoking the 'Exit' command prompts for a session filename prior to ending the program (if a session has been initiated). Responding to this prompt is optional, and pressing <Enter> without an entry or pressing the <ESC> key causes the program to exit without saving the session.

10.3. Preference file

Preferences set up in the IGS ('Graph Options' and 'Preferences' screens) are recorded in a file called 'PREFile.WKS'. This file is loaded automatically when the CPP is executed, and is saved automatically on program exit. To begin with the default set of preferences, 'PREFile.WKS' may be erased prior to running the CPP program.

10.4. Dependencies

This section should be read by programmers who wish to recompile the CPP program (i.e. with/without MATH chip support). The CPP consists of a main program (PLT.PAS), a file loading unit (PLTFile.PAS), the IGS units (see Table 7.3, p. 73), and the CGP units (Table 7.3). To recompile with a MATH chip support option different than the current setting, all CPP, IGS, and CGP units must be recompiled with the new setting (the "Build" function of TP will do this if started when PLT.PAS is loaded).

Chapter 11 - CPP DATA FILE FORMAT

The CPP reads ASCII data files. This approach allows much flexibility in the data sources since files can be generated and manipulated easily by many different utilities prior to being loaded into the CPP. Three record formats are recognized by the CPP. These include a data record, a field display format record, and a column title record. If a display format record is used, it must be the first record in the data file. Then the column title record must follow (if it is used), and finally all the data records.

11.1. Data records

Data files used by the CPP contain a record (text line) for every row of data field values. Each row may contain numeric and alphanumeric fields, separated by spaces and/or commas. All fields beginning with a number are interpreted as numeric and are converted to real. All fields beginning with a non-numeric character are interpreted as alphanumeric fields and are stored unchanged. Data can also be enclosed in single or double quotes to force alphanumeric interpretation, and/or to allow multiple word fields. Missing fields can be represented by two commas with no data between them. An example of a data record is shown here:

```
35, -16, -73.02, , salmon, '1988 Review', 25
```

The first data record encountered determines the number of columns in the data-set. The type (numeric or alphanumeric) of each column encountered in the first data record must be consistent in all consecutive data records. Also, each data record must have the same number of fields (columns). The first blank or empty record, or end-of-file encountered, marks the end of the data-set.

11.2. Field display format record

In addition to data records, a special record containing display format information may also be included in a data file. The display format for each column is specified by a record beginning with a '%' character, then followed by a series of 'M:N' format number pairs separated by commas and/or spaces. Each number pair corresponds to a data column. More or fewer format pairs may be specified than data columns, but missing fields (double commas with nothing between) are not allowed. An example of a field display format record is shown here:

```
% 10:0, 10:2, 12:-1   10:2
```

The first number 'M' determines the total width of the field, and the second number 'N' determines either the precision, or the display format. Alphanumeric fields make use of only the 'M' value. Text is truncated to a width of 'M' characters. Numeric fields make use of the 'N' value as specified in Table 7.1 (p. 69).

When a data file is loaded, and a display format record is specified, the largest 'M' value is found and the cell width of the spreadsheet is set to the largest 'M' + 1. If not enough 'M:N' pairs are provided for all columns, free format is used (N = -2) for the remaining columns, with 'M' set to the cell width less one.

11.3. Column title record

Titles may be specified for each column by including a column title record. This consists of a '~' character followed by a list of alphanumeric fields separated by spaces and/or commas. Each alphanumeric field corresponds to a column. Missing column titles are allowed. This is an example of a column title record:

```
~ 'Species', Length, , 'Wgt (gm)', Total
```

PART FOUR

MRP
GRAPHICS
SYSTEM
(MRPGS)

Chapter 12 - MRPGS INTRODUCTION
Chapter 13 - MRPGS USER MANUAL

Chapter 12 - MRPGS INTRODUCTION

The Mark Recovery Program Graphics System (MRPGS) provides plotting capabilities for users of the Mark Recovery Program (MRP) database. Readers unfamiliar with the MRP database should consult the report by Kuhn et al. (1988) which offers a detailed description of the database.

To use the MRPGS, one must first create MRP flat data files (Kuhn 1988) for the MRPGS to process. When the MRPGS is run, a screen is displayed, and the user must fill in certain fields to instruct the MRPGS file processor to create a summary data spreadsheet from the input data files. It is from this summary spreadsheet that graphs are produced.

12.1. Brief overview of MRP

The Mark Recovery Program is a program to monitor and assess salmon enhancement efforts using tagging techniques. Tag releases and tag recovery information reside in a computer database on a VAX minicomputer system at the Pacific Biological Station in Nanaimo B.C.

A flexible report writing package, the MRP-Reporter (Kuhn 1988), is available for data extraction from the MRP database. ASCII data files containing all data fields that comprise a release and recovery data record can be produced using the MRP-Reporter program. These files may then be transferred from the VAX minicomputer to a microcomputer for import into the MRPGS.

12.2. The input data files

Two data files are required as input to the MRPGS. They both must have the same name and use the file extensions: ".REL" and ".REC". The MRP-Reporter program provides a function to create these data files using data extracted from the MRP database. This function is selectable from the MRP-Reporter's System Access menu via the item titled "Flat File builder".

These two data files must be sorted in ascending order using the first 12 characters (the tag code) of each record. This requirement must be adhered to, or the MRPGS file processing function may produce erroneous results. Furthermore, the record format of these files must not be tampered with as the MRPGS expects it to be exactly the same as that defined by the MRP-Reporter program.

12.3. Relationship to the IGS

Graphics production is performed by the IGS component of the CGP. The MRPGS is simply a custom made front-end to the IGS. It processes some data files into the format required by the IGS and then hands control over to the IGS. Upon termination of the IGS, control is returned to the MRPGS which allows the user to save the session and exit to DOS.

Chapter 13 - MRPGS USER MANUAL

13.1. Overview

A three stage process is involved in producing graphics from MRP data. First, the user must run the MRP-Reporter program to specify the data requirements and create the two required data files. Second, the two data files must be transferred to a microcomputer. Lastly, the MRPGS (MRPlot.EXE) program is run and the name of the two data files is provided by the user. At this point the user fills in a few fields from the MRPGS screen, instructs the MRPGS to process the data, and then uses the IGS to set up data plots.

Two technical reports by Kuhn et al. (1988) and Kuhn (1988), are required reading for users not familiar with the MRP database and the MRP-Reporter. The remainder of this chapter assumes users are comfortable with MRP database jargon and concepts.

13.2. Main Menu

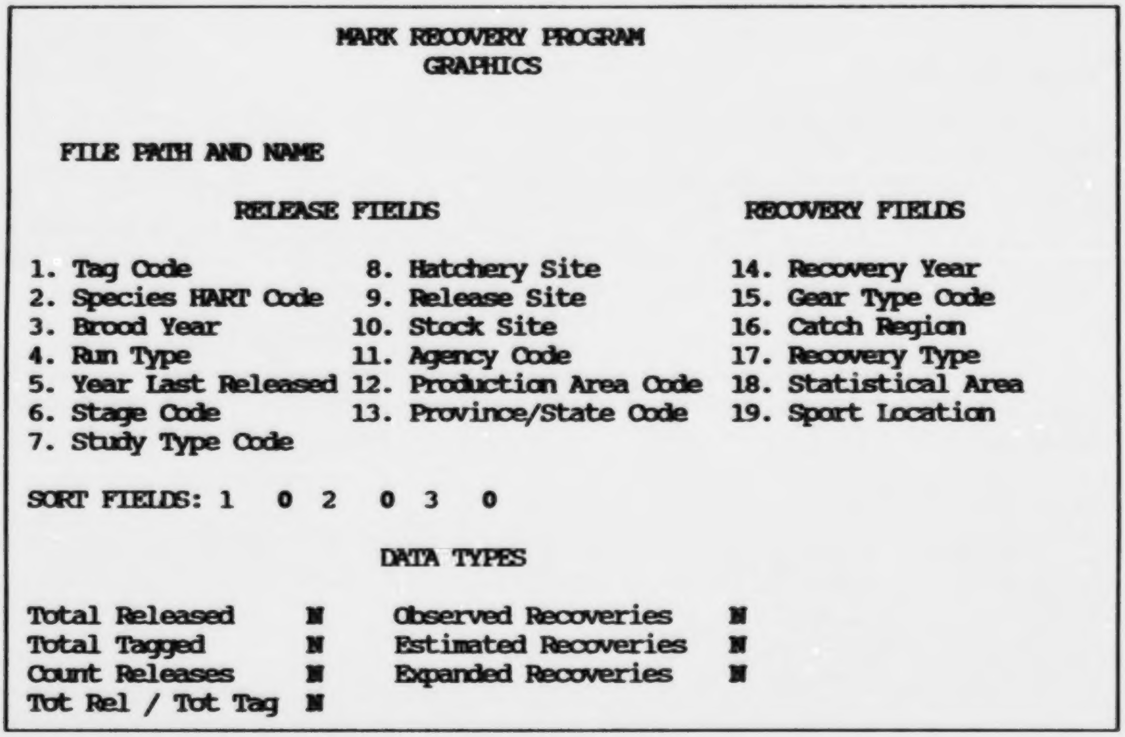
When the MRPGS is first executed it asks the user whether it should load previously defined definitions (actually IGS session files):

Load definitions ? [Y/N]

[If the user responds with a "N", the program clears the screen and displays the Main Menu (see Figure 13.1, p. 104). A "Y" response causes the program to prompt for the name of a definitions (session) file (see the next prompt.]

FileName(no extension)?

[This prompt may be abandoned by pressing the <ESC> key or an existing definitions filename may be entered (see section 7.3.6 for details on session files).]



```
                         MARK RECOVERY PROGRAM
                               GRAPHICS

  FILE PATH AND NAME

               RELEASE FIELDS                       RECOVERY FIELDS

1. Tag Code              8. Hatchery Site          14. Recovery Year
2. Species HART Code     9. Release Site           15. Gear Type Code
3. Brood Year           10. Stock Site             16. Catch Region
4. Run Type             11. Agency Code            17. Recovery Type
5. Year Last Released   12. Production Area Code   18. Statistical Area
6. Stage Code           13. Province/State Code    19. Sport Location
7. Study Type Code

SORT FIELDS: 1   0 2   0 3   0

                         DATA TYPES

Total Released     N    Observed Recoveries    N
Total Tagged       N    Estimated Recoveries   N
Count Releases     N    Expanded Recoveries    N
Tot Rel / Tot Tag  N
```

Figure 13.1 - MRP Graphics System: Main Menu

13.2.1. Layout and description

The first field **FILE PATH AND NAME**, must be filled in with the name of the MRP flat files to be processed. This field is filled in automatically if a session was loaded.

Nineteen different data fields may be used as levels in the data sorting process. Sort levels are hierarchical, i.e., level-three sorts within level-two, and level-two sorts within level-one. Certain data fields do not make sense as lower sort levels for other data fields (e.g., **Hatchery Site** within **Tag Code**). However, the program allows any combination, and it is therefore the user's knowledge of the MRP database structure that will assist in sensible sort level combinations. To select a particular data field for a sort level, move the cursor into the appropriately **SORT FIELDS:** numbered field (1, 2, or 3) and type in the data field number.

Finally, the user must select one or more (up to seven) data types for the MRPGS to compute. However, there is one restriction, the program does not allow release data types to be select with recovery sort fields. To select a data type, move the cursor into the desired **DATA TYPE** field, and change the rotary field from "N" to "Y" (rotary fields are changed by pressing <Ctrl-Right-Arrow> or <Ctrl-Left-Arrow>).

13.2.2. Processing the data files

Pressing the <ESC> key after the Main Menu input screen is filled in, causes the MRPGS to begin processing the data files. The data are summarized into a spreadsheet format based on the sort levels and data types specified.

When processing commences, the program displays status information at various stages of execution. After the processing has completed, the program calls the IGS. An example session would go something like this:

1. Run the MRP-Reporter program to create the data files selecting:
    - COHO and CHINOOK
    - 1979-80 brood years
    - Nanaimo River CDP and Conuma River hatcheries
    - Production releases only
    - 1982-84 recovery years

   Name the flat files "TESTSET"

2. Transfer TESTSET.REL and .REC to a microcomputer

3. Run the MRPGS program: "MRPlot"
    - Choose 'Tag Code' for sort level one
    - Choose data types: 'Total Released' and 'Observed Recoveries'
    - Press the <ESC> key to begin processing

The processing status:

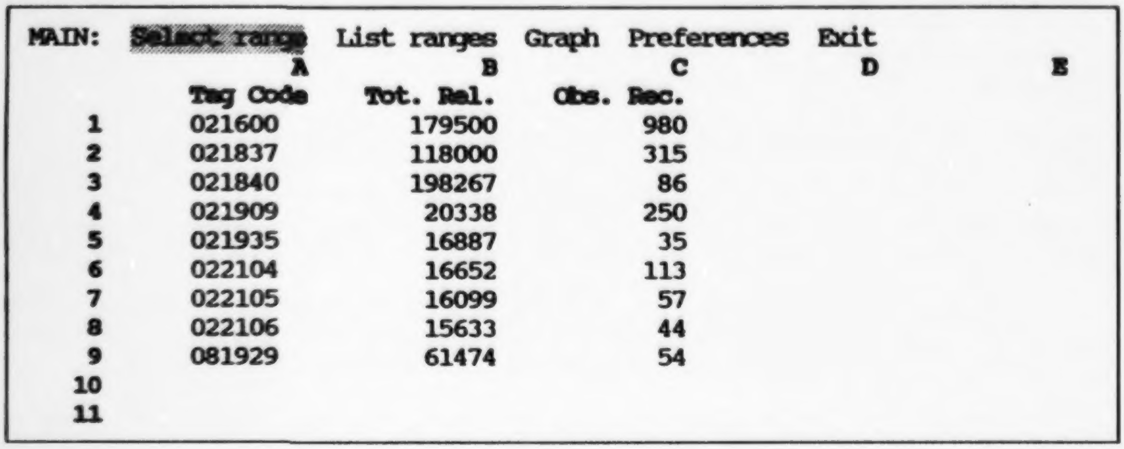
```
Release file required
Recovery file required
Number of sort fields : 1
Number of data fields : 2
Start building sort field symbol table
```

and:

```
Number of release records :  9
Number of recovery records : 1661
Size of spreadsheet array, rows: 9 cols: 3
Start accumulating data
```

(During processing, the MRPGS displays dots "." across the screen to indicate that the program is still executing.)

The program passes control to the IGS, which displays the summary spreadsheet in the IGS Main Menu (Figure 13.2).



```
                    A               B              C              D              E
             Tag Code       Tot. Rel.      Obs. Rec.
     1         021600          179500            980
     2         021837          118000            315
     3         021840          198267             86
     4         021909           20338            250
     5         021935           16887             35
     6         022104           16652            113
     7         022105           16099             57
     8         022106           15633             44
     9         081929           61474             54
    10
    11
```

Figure 13.2 - MRP Graphics System example IGS spreadsheet

By choosing the 'Exit' command from the IGS Main Menu, program control is returned to the MRPGS which responds with:

Do another set ? [Y/N]

[Enter a "Y" to return to the MRPGS Main Menu or an "N" to enter the program termination phase (see the next prompt).]

Save definitions ? [Y/N]

[Enter an "N" to terminate the program or a "Y" to save the session (see the next prompt).]

FileName?

[Enter the name of the session file to save the IGS Graph definitions to or press <Enter> without a name to abandon the save operation.]

13.3. Spreadsheet format

Sort fields occupy the first 1 - 3 columns in the spreadsheet, and data fields the remaining 1 - 7 columns. In Figure 13.2, the three columns consist of one sort field (Tag Code) and two data types (Tot. Rel. and Obs. Rec.). Tables 13.1 and 13.2 (p. 107) list the sort field and data type column names used by the MRPGS.

Table 13.1 - List of Sort field Menu / Column names

Menu names	Column names
Tag Code	Tag Code
Species HART Code	Species
Brood Year	Brood Yr
Run Type	Run Type
Year Last Released	Yr Lst Rel
Stage Code	Stage Cd
Study Type Code	Study Cd
Hatchery Site	Hatchery
Release Site	Rel. Site
Stock Site	Stock Site
Agency Code	Agency Cd
Production Area Code	Prod. Area
Province/State Code	Prov/State
Recovery Year	Rec. Yr
Gear Type Code	Gear Type
Catch Region	Catch Reg.
Recovery Type	Rec. Type
Statistical Area	Stat. Area
Sport Location	Sport Loc.

Table 13.2 - List of Data field Menu / Column names

Menu names	Column names
Total Released	Tot. Rel.
Total Tagged	Tot. Tag.
Count Releases	No. Rel.
Tot Rel / Tot Tag	Rel / Tag
Observed Recoveries	Obs. Rec.
Estimated Recoveries	Est. Rec
Expanded Recoveries	Exp. Rec.

All sort fields, excluding those which are years (i.e. Brood Year, Recovery Year, and Last Year Released), are stored and presented as character strings, and may only be used for specifying categories in discrete-continuous plots (bar charts). Years and data field columns are real numbers, and may be used either as discrete variables or as continuous variables (this dual interpretation is described in the IGS manual).

The data in the spreadsheet are sorted by row according to the order of specification of the sort fields. The sort field, entered first on the input screen, occupies the leftmost column and varies least rapidly.

13.4. Dependencies

The MRPGS consists of a main program file (MRPlot.PAS), a screen processing unit (MRPScrn.PAS), a file processing unit (MRProc.PAS), the IGS units and the CGP units (see Table 7.3, p. 73). To recompile with a MATH chip support option different than the current setting, all MRPGS, IGS, and CGP units must be recompiled with the new setting (the TP Build function may be used for this).

PART FIVE

IMAGE
PRINTING
(PRT)

Chapter 14 - PRINTIMAGE DESCRIPTION
Chapter 15 - PRINTIMAGE USER MANUAL

Chapter 14 - PRINTIMAGE DESCRIPTION

Printing graphic images on hard copy devices is a straightforward process. However, a good printing program should have fast execution speed, flexible image size, rotation capability, and position control. The PrintImage program addresses these concerns.

14.1. Introduction

PrintImage is implemented in three executable programs and a linkable TP unit. Each of the three executable programs detect and support different graphics cards while the TP unit version supports and detects the presence of three different graphics cards.

The three executable versions are Terminate and Stay Resident (TSR) modules. A TSR is a program that is loaded into computer memory to co-exist with other programs. To invoke a TSR, a "hot key" sequence is issued by the user to "pop up" the TSR, effectively suspending the current program's execution and transferring control to the TSR. When the TSR terminates (but remains available), it returns control to the current program. The current program does not know that it was suspended and will simply continue executing. This capability allows PrintImage to be used with a graphics display created by an arbitrary program. Each TSR version has a different file name dependent on the graphics card supported:

PRTHerc.EXE	► Hercules Monochrome Graphics Card version ◄
PRTCGA.EXE	► Color Graphics Adapter version ◄
PRTEGA.EXE	► Enhanced Graphics Adapter version ◄

These TSRs allocate approximately 38 kilobytes (K) of the computer's RAM memory. This memory is not available for use by other programs. When a TSR version is removed from memory, the 38 K block is released.

An alternative to the TSR version is the TP linkable unit version, callable from a TP program by a procedure reference. The name of this unit version of PrintImage is:

PRT.TPU

Various options are provided for controlling the orientation and size of the printed picture. Users may instruct PrintImage to rotate the picture 90°. The picture may be stretched in width or height. Different printer resolutions are selectable. PrintImage provides control of these options via menu prompts, or in the case of the unit version, optional procedure parameters.

14.2. Graphics card and mode support

Currently, three graphics cards are recognized and supported by PrintImage. They are:

- Hercules Monochrome Text/Graphics Card (HGC)
- Color Graphics Adapter (CGA)
- Enhanced Graphics Adapter (EGA) [with 128 KBytes of on-board RAM]

A fourth graphics card, the Virtual Graphics Array (VGA), will be supported in a future release of PrintImage. Most graphics cards support one or more of these graphics standards (some offer all modes from VGA down to HGC).

To reduce memory requirements, each card is supported by a different TSR module. Each module, when activated, attempts to detect the card it is designed to support. If detection is unsuccessful, the program emits a low buzz to indicate this. Only certain modes for each card type are supported (Table 14.1).

Table 14.1 - List of Graphics cards and modes supported by PrintImage

Graphics card	BIOS Mode	Description
HGC	7*	720x348 monochrome graphics
CGA	4	320x200 four color
	5	320x200 four color (grey scale)
	6	640x200 two color (B&W)
EGA	16	640x350 64 color (16 active colors)

[* (HGC) Graphics mode is internal to the card; the BIOS views text mode as always active.]

The unit version of PrintImage will detect any of the three cards listed and will configure itself accordingly.

14.3. Printer support

Two types of printers are currently supported, the Epson dot matrix standard and the Hewlett-Packard LaserJet Series II standard. Most Epson printer modes and all of the HP LaserJet modes are supported (see section 15.2.4). Epson users must ensure that the printer is 100% Epson compatible, otherwise unpredictable results will occur. HP LaserJet users should be aware that graphics printing requires a significant amount of internal printer memory. Certain combinations of image expansion and high resolution printer modes may not be compatible.

14.4. Dependencies

PrintImage is implemented without reference to the TP Graph unit or the CGP family of units. Graphics detection, scanning, and display routines are written with specific reference to the 6845 display controller chip that resides on each of the three supported graphics cards (some BIOS functions are utilized). This imposes a significant level of hardware dependency, but offers a much welcomed boost in performance.

To create the TSR versions, a set of third-party TP utilities is employed (TurboPower 1988). These utilities are incorporated by referencing various procedures and functions exported by two custom TP units provided by TurboPower Software. If users wish to modify the TSR versions of PrintImage, they must purchase a copy of these utilities.

To incorporate PrintImage into a custom program, the programmer must include a reference to the PRT unit in the USES statement of their program, as shown here:

```
USES PRT;
```

Chapter 15 - PRINTIMAGE USER MANUAL

15.1. Installing/Calling PrintImage

PrintImage is available as a stand-alone executable program and as a linkable TP unit file. The stand-alone version comes as three different executable files, one for each supported graphics card. By contrast, the linkable unit contains one entry point which determines the graphics card in use and configures itself accordingly. Starting PrintImage in these different ways is discussed in the next two subsections.

15.1.1. Installing the TSR version of PrintImage

Since PrintImage is broken into three separate executable programs (one for each supported graphics card), the user must decide which graphics card he will be using. For the remainder of this section we will refer to the EGA graphics card.

To install PrintImage, the user must enter the command PRTEGA at the DOS prompt (see section 14.1 for CGA and Hercules versions). The program file PRTEGA.EXE must be accessible from the current DOS path. If all goes well, the message

```
PrintImage EGA v1.0 loaded:
Press <Ctrl><RightShift><PrtScr> to activate.
     [Must be in graphics mode!!!]
Enter PRTEGA/h for more information.
Enter PRTEGA/r to REMOVE.
```

will appear, indicating that the program is now resident in memory. If for some reason it can not go resident, this message will appear immediately following the above dialog:

```
Unable to go resident!!!
```

To actually activate the PrintImage program, the user must display a graphic image, and enter the "hot key" sequence <Ctrl><RightShift><PrtScr>.

If the user has already installed PrintImage and attempts to run the PRTEGA program again, PrintImage will respond with this message:

```
PrintImage v1.0 already installed!!!
```

Issuing the command PRTEGA/h (or PRTEGA/H) at the DOS prompt will display a brief help screen to remind the user of the commands and features provided.

15.1.2. Calling the TP unit version of PrintImage

Users may include a call to PrintImage inside their own TP programs. This is the TP procedure definition:

```
Procedure PrintImage(OptionCode : CHAR;
                     VAR PrnType,PrnPort,ScanType,PrnMode : CHAR;
                     VAR WidthFactor,HeightFactor : REAL;
                     VAR ResultCode : BYTE);
```

The OptionCode parameter determines whether PrintImage should reference the next six parameters or collect their values by interacting with the user via the PrintImage Menu. Refer to Table 15.1 (p. 115) for a list of valid parameter values. An example call to **PrintImage** would be as follows:

```
                    .
                    .
                    .

Var   PrnType,PrnPort,ScanType,PrnMode : CHAR;
      Width,Height : REAL;  ErrCode : BYTE;
                    .
                    .
                    .
Begin
                    .
                    .
                    .
  PrtType := 'E';         { select Epson dot-matrix }
  PrnPort := '0';         { select parallel port LPT1 }
  ScanType := 'P';        { select Landscape page orientation }
  PrnMode := '1';         { select Double speed/Double density }
  Width := 1.5; Height := 1.5;

            { Use parameters only - no menu }
  PrintImage( 'P',PrnType,PrnPort,ScanType,PrnMode,
             Width,Height,ErrCode );
  If (ErrCode > 0) Then WriteLn('* Error no. ',ErrCode);
                    .
                    .
                    .
End.
```

Table 15.1 - PrintImage procedure parameter values

Parameter	Description and accepted value(s)
OptionCode	P (use parameters), M (use menu), or B (use pars. and menu)
PrnType	Printer Type: E (Epson) or H (HP LaserJet Series II)
PrnPort	Printer Port: 0-2 (LPT1-LPT3)
ScanType	Scan Type: P (portrait) or L (landscape)
PrnMode	Printer Mode: 1-4 (depends on printer type) **Epson:** 1 - Double Density Double Speed 2 - Quad Density 3 - Standard Density 4 - Double Density **HP:** 1 - 75 Dots per inch (DPI) 2 - 100 DPI 3 - 150 DPI 4 - 300 DPI
WidthFactor	Image width multiplication factor. A value of 1.0 prints a one-to-one screen-to-printer image width.
HeightFactor	Image height multiplication factor. A value of 1.0 prints a one-to-one screen-to-printer image height.
ResultCode	Returned to caller indicating the success or failure of the task. These values are returned: 0 - Successful completion 1 - Invalid OptionCode value 2 - Invalid PrnType value 3 - Invalid PrnPort value 4 - Invalid ScanType value 5 - Invalid PrnMode value 6 - Invalid WidthFactor value 7 - Invalid HeightFactor value 8 - Printer ERROR (off-line etc.) 9 - Interrupted by the user (pressed <ESC> key)

The **ResultCode** parameter is returned with a value of 0-9. Each parameter is tested in succession. For example, if the **OptionCode** and **PrnPort** values are okay but not the **ScanType** value, a **ResultCode** of 4 would be returned.

15.2. The PrintImage Menu

When PrintImage "pops up", it beeps, clears the bottom 10 scan lines, and spins a dashed rectangle around the blinking text "PrintImage v1.0". Depending on the graphics card and mode, color is applied.

> [Hercules Graphics Card users: if the computer beeps but does not perform as described above, press the <PgUp> key to instruct PrintImage to re-active itself on the alternate graphics page (this card has two graphics memory pages). A program cannot detect which page is currently active.]

The menu can be entered at this point (hit <Enter> or <Space>) or the user may choose to accept default settings and bypass the menu. There are two kinds of defaults: "hard-wired" and "last-run". Hard-wired defaults are preprogrammed default menu prompt choices/values. On the other hand, last-run defaults are the menu prompt choices/values that were changed or accepted by the user earlier in the session. The hard-wired defaults are used for last-run defaults on the first activated session. Table 15.2 below lists and describes the keystrokes available for interacting with PrintImage.

There are six different menu prompts. The first four require non-numeric responses requiring a choice from a list of alternatives. The last two require numeric values as input. Both selections and numeric values are not explicitly entered by the user, but are selected via the arrow keys (←↓↑→), as explained in Table 15.2. Pressing the <Enter> key for any prompt accepts the choice or value given.

If the printer is OFF-LINE or has a problem, the program will display this dialogue prior to entering the menu:

```
Printer Error !!!           --- pause ---

Please Correct !!!          --- pause ---

<ESC> to Abort or <Enter> to Resume
```

Pressing the <ESC> key will abandon the session and pressing the <Enter> key will cause the program to test the printer again.

Table 15.2 - PrintImage Menu valid keystrokes

Keystroke	Description
<ESC>	At any time during a session with PrintImage, the user may press the <ESC> key to abort the session and return to the point of entry.
<Enter>	Accepts the choice/value offered by the program at a menu prompt.
<F1>	The program will set all remaining menu prompt choices/values to the "last-run" settings and proceed to the image printing task.
<F2>	Same as <F1> except the program will use "hard-wired" settings.
↑ <Up-Arrow>	If the prompt is non-numeric then cycle to the previous choice list entry else add one to the left-most digit.
↓ <Dn-Arrow>	If the prompt is non-numeric then cycle to the next choice list entry else subtract one from the left-most digit.
← <Left-Arrow>	If the prompt is non-numeric then same as <Up-Arrow> else subtract one from the second digit.
→ <Right-Arrow>	If the prompt is non-numeric then same as <Dn-Arrow> else add one to the second digit.
<Ctrl-Right-Arrow>	If the prompt is numeric then add one to the third digit.
<Ctrl-Left-Arrow>	If the prompt is numeric then subtract one from the third digit.
<Home>	If the prompt is non-numeric then cycle to the "hard-wired" default choice setting else set value to 1.0.

15.2.1. Printer types

Printer type selection is provided by this prompt:

```
PRINTER TYPE : Epson                ► default ◄
               HP LaserJet
```

Most dot matrix printers support the Epson graphics standard, therefore dot matrix printer users should select **Epson** as the printer type. **HP LaserJet** should be selected by Hewlett-Packard LaserJet Series II users. Graphics printing with the HP LaserJet requires a significant quantity of printer memory. This depends on the amount of graphics data to print (i.e., size and no. of dots-per-page) and the printer mode selected (see section 15.2.4).

15.2.2. Selecting printer ports

The second menu prompt to appear is:

```
PRINTER PORT : LPT1                ► default ◄
               LPT2
               LPT3
```

This function allows the user to select the printer port to use. (Incidentally, LPT1 is usually equivalent to PRN.)

15.2.3. Scan types

Scan type selection is available from this prompt:

```
SCAN TYPE : Portrait               ► default ◄
            Landscape
```

A scan type is the direction the program scans the graphics display. A Portrait scan is left to right/top to bottom and a Landscape scan is bottom to top/left to right. This orientation is reflected on the printed page.

15.2.4. Printer modes

Users of Epson dot matrix printers must ensure that the printer's automatic line feed dip switch is turned off. Printer mode selection is provided by this prompt:

```
[ Epson ]

PRINTER MODE : Dbl Dnsty Dbl Spd   ► default ◄
               Quad Dnsty
               *Standard Dnsty
               Dbl Dnsty
```

* Standard Dnsty may not be available. Certain screen resolution - scan type combinations are not possible.

```
[ HP LaserJet ]

PRINTER MODE : 75 DPI         ► default ◄
               100 DPI
               150 DPI
               300 DPI
```

Changing printer modes changes the size of the printed image. For example an image printed in a high resolution mode such as **Quad Dnsty** (or 300 DPI for the **HP LaserJet**) with scaling factors equal to 1.00 results in a small picture. To increase the size of the printed picture, users may decrease the printing resolution by selecting a different printer mode (e.g. **Dbl Dnsty** for the **Epson** or 100 DPI for the **HP LaserJet**).

15.2.5. Scaling factors

Scaling factors range from 1.00 to 9.99. Valid ranges depend on the printer type and mode. A factor of 1.0 will produce an image with a one-to-one mapping with the screen image. Factors greater than 1.0 increase the printed image size by that factor. Support for factors less than one is not provided. Default factors are offered by the program, which attempt to fit the image to an 8½ x 11 inch page (Epson only) while preserving the screen's aspect ratio. These factors reference the screen not the printed page, so that the width factor increases the width of the screen image when printed. There are two scaling factors provided:

WIDTH SCALE FACTOR : n.nn

and

HEIGHT SCALE FACTOR : n.nn

The order of these two prompts is reversed if the scan type **Landscape** is selected.

For the HP LaserJet printer, increasing the image size can dramatically increase printer memory requirements, because laser printers require a page to be completely composed prior to printing. A larger image size may be achieved via a lower dot-per-inch mode selection (see section 15.2.4 above).

15.3. Removing the TSR version of PrintImage from memory

With the proliferation of TSR programs, users must have the ability to remove a TSR program when it is not required. PrintImage allows the user to remove it by issuing this command at the DOS prompt:

```
PRTEGA/r          ► or PRTCGA or PRTHerc depending on graphics card ◄
```

Certain conditions, such as another TSR loaded after PrintImage, could prevent it from removing itself. If this is the case, the program displays this message:

```
UNABLE To SAFELY REMOVE PrintImage!!!
```

or, if the remove request is successful:

```
PrintImage REMOVED!!!
```

or, if PrintImage is not installed:

```
PrintImage is not installed.
```

Appendix A - TECHNICAL SPECIFICATIONS

In this appendix we recommend a minimum system configuration and provide information on graphics hardware programming, memory requirements, and Turbo Pascal version 4.0 and 5.0 compiler settings. Programmers should find the discussion of graphics hardware programming interesting.

■ Minimum System Configuration

IBM PC/AT/386 or compatible
at least 512 KBytes RAM
Text/Graphics Card*
Video Monitor compatible with the graphics card
1 360 KByte floppy
10 MByte or larger fixed disk

* One of: Hercules Graphics Card or Color Graphics Adapter or Enhanced Graphics Adapter or Compaq Plasma. Support for other cards (such as VGA) may be supported in future releases of the software.

■ Memory Requirements

Custom Graphics Package

Memory usage depends a great deal on what functions the programmer uses from the CGP. For example, a call to the routine **IGSMainMenu** links in most of the CGP functions and procedures as well as most IGS specific utilities. A very simple program that calls **IGSMainMenu** will reside in an executable file occupying approximately 170 KBytes. The memory requirement would be 170 KBytes plus data and stack allocations.

Image Printing

A PrintImage TSR program requires approximately 38 KBytes of memory. This memory is allocated for the remainder of the TSRs residency.

■ Turbo Pascal version 4.0 and 5.0 compiler settings

Certain compiler settings can have a significant effect on program performance and executable size. Some settings must be the same for all modules. Listed here are the settings used to compile the CGP (includes the IGS, CPP, MRPGS, and demo programs) and PrintImage. These settings are accessible from the Options menu item in the Turbo Pascal interactive environment main menu.

Custom Graphics Package

Range checking	On
Stack checking	On
I/O checking	On
Debug information	On
Turbo pascal map file	Off
Force far calls	Off
Var-string checking	Relaxed
Boolean evaluation	Short-circuit
Numeric processing	Hardware or Software
Emulation	On or Off
Link buffer	Disk
Memory sizes	
Stack size	16384
Low heap limit	0
High heap limit	655360

Image Printing Unit

* Same as above except for:

Range checking	Off
Stack checking	Off
I/O checking	Off
Numeric processing	Software
Memory sizes	
Stack size	2048
High heap limit	0

Appendix B - PROCEDURE/FUNCTION REFERENCE

This appendix comprises an alphabetic list of all plotting services exported by the KPS, BPS, 2DPS, 3DPS, IGS, and PRT units. Listed are the service names and parameter lists, the unit that implements it, and the page number on which it is documented.

Service names and parameter lists	Unit	Page no.
AccumulateHistogram(VAR X,Y,A; N,Axis,Ints : WORD);	2DPS	48
Axis(AxisName : WORD; Min,Max : REAL; OptionStr,LabelStr : STRING);	BPS	30
AxisD(NyCats : WORD; VAR CatLabels; LabelStr : STRING);	BPS	31
CharDepth(FontUse,Direction : WORD) : REAL;	BPS	36
CharHeight(FontUse,Direction : WORD) : REAL;	BPS	36
Contour(Nx,Ny : WORD; VAR X,Y,Z; Nc : WORD; VAR Zc; Options : STRING; PosColor,ZeroColor,NegColor : WORD);	2DPS	52
CornerBox(Options : STRING; NLegend,NHeader : WORD);	3DPS	59
DefineRegion(Nx,Ny : WORD);	KPS	20
DefineViewport(I : INTEGER; Rx1,Ry1,Rx2,Ry2 : REAL);	KPS	20
DefineWorld(I : INTEGER; Rx1,Ry1,Rx2,Ry2 : REAL);	KPS	21
DrawVPBox;	KPS	21
HardCopyOff;	KPS	19
HardCopyOn;	KPS	19
IGSMainMenu(DataArray : POINTER; Rows,Cols : INTEGER; ColTypeArray : DataFieldTypeArrayPtr; ColFmtArray : ColFmtArrayPtr; ColTitleArray : ColTitleArrayPtr);	IGS	66
IGSRelease;	IGS	69
InitCGP(VAR GraphDriver,GraphMode : INTEGER; DriverPath : STRING);	BPS	29
InitKPS(VAR GraphDriver,GraphMode : INTEGER; DriverPath : STRING);	KPS	17
LabelLolliPoints(N : WORD; VAR X,Y,Z,Labels; Location : WORD);	3DPS	60
LabelPoints(N : WORD; VAR X,Y,Labels; Location,YAxisName : WORD);	2DPS	51
LoadDriver(FileString : STRING) : BOOLEAN;	KPS	19
LoadFont(FileString : STRING) : BOOLEAN;	KPS	19
LoadPrefs(Filename : STRING);	IGS	70
LoadSession(SessionFilename : STRING) : BOOLEAN;	IGS	70
Log10(X : REAL) : REAL;	BPS	32
Lolliplot(N : WORD; VAR X,Y,Z; Symbol,LegendNum : WORD; LegendStr : STRING);	3DPS	59
MainHeader(HString : STRING);	BPS	37
MapFont(FontUse,FontName : WORD; FactorWidth,FactorHeight : REAL);	BPS	33
MaxInArray(N : WORD; VAR A) : REAL;	BPS	32
MinInArray(N : WORD; VAR A) : REAL;	BPS	32
NArc(Rx,Ry,Radius : REAL; StAngle,EndAngle : WORD);	KPS	25
NBar(Rx1,Ry1,Rx2,Ry2 : REAL);	KPS	24
NBar3D(Rx1,Ry1,Rx2,Ry2,Depth : REAL; Top : BOOLEAN);	KPS	24
NCircle(Rx,Ry,Radius : REAL);	KPS	26
NClearViewPort;	KPS	21
NDrawPoly(NumPoints : WORD; VAR RPolyPoints);	KPS	25
NEllipse(Rx,Ry,Xr,Yr : REAL; StAngle,EndAngle : WORD);	KPS	26
NextRegion;	KPS	20
NFillPoly(NumPoints : WORD; VAR RPolyPoints);	KPS	25
NFloodFill(Rx,Ry : REAL; Border : WORD);	KPS	25
NGetArcCoords(VAR AC : NArcCoordsType);	KPS	25
NGetPixel(Rx,Ry : REAL) : WORD;	KPS	22

Service names and parameter lists	Unit	Page no.
NGetX : REAL;	KPS	22
NGetY : REAL;	KPS	22
NLine(Rx1,Ry1,Rx2,Ry2 : REAL);	KPS	23
NLineRel(Rdx,Rdy : REAL);	KPS	23
NLineTo(Rx,Ry : REAL);	KPS	23
NMoveRel(Rdx,Rdy : REAL);	KPS	22
NMoveTo(Rx,Ry : REAL);	KPS	22
NOutText(TextString : STRING);	KPS	26
NOutTextXY(Rx,Ry : REAL; TextString : STRING);	KPS	26
NPieSlice(Rx,Ry,Radius : REAL; StAngle,EndAngle : WORD);	KPS	26
NPutPixel(Rx,Ry : REAL; Pixel : WORD);	KPS	22
NRectangle(Rx1,Ry1,Rx2,Ry2 : REAL);	KPS	24
NSetLineStyle(LineStyle,Pattern,Thickness : WORD);	KPS	24
NSetUserCharSize(FactorX,FactorY : REAL);	KPS	26
NTextHeight(TextString : STRING) : REAL;	KPS	27
NTextWidth(TextString : STRING) : REAL;	KPS	27
OutAText(TextStr : STRING);	BPS	35
OutATextXY(X,Y : REAL; TextStr : STRING);	BPS	35
OuterFrame;	BPS	38
PlaceSymbol(X,Y : REAL; Symbol : WORD);	BPS	38
PlaneBox(Options : STRING; NLegend,NHeader,NyAxes : WORD);	2DPS	43
PlotBar(N : WORD; VAR X,Y; YAxisName,Pattern,LegendNum,TotBars,BarNum : WORD; LegendStr : STRING);	2DPS	45
PlotBarD(VAR Data; BDT : BarDataType; Pattern,LegendNum : WORD; LegendStr : STRING);	2DPS	47
PlotBarS(VAR Data,SLabels,BarColors; BDT : BarDataType; Options : STRING);	2DPS	47
PlotHistoBar(NBin : WORD; VAR Y; YAxisName,Pattern,LegendNum : WORD; LegendStr : STRING);	2DPS	50
PlotHistoLine(NBin : WORD; VAR Y; YAxisName,LineStyle,LegendNum : WORD; LegendStr : STRING; LineOption : WORD);	2DPS	49
PlotHistoSymbol(NBin : WORD; VAR Y; YAxisName,Symbol,LegendNum : WORD; LegendStr : STRING);	2DPS	50
PlotLine(N : WORD; VAR X,Y; YAxisName,LineStyle,LegendNum : WORD; LegendStr : STRING);	2DPS	44
PlotSymbol(N : WORD; VAR X,Y; YAxisName,Symbol,LegendNum : WORD; LegendStr : STRING);	2DPS	45
PlotSymbolD(VAR Data; BDT : BarDataType; Symbol,LegendNum : WORD; LegendStr : STRING);	2DPS	46
PrintError;	KPS	17
PrintImage(OptionCode : CHAR; VAR PrnType,PrnPort,ScanType,PrnMode : CHAR; VAR WidthFactor,HeightFactor : REAL; VAR ResultCode : BYTE);	HRT	114
RecordError(ErrStr : STRING);	KPS	17
ResetError;	KPS	17
SavePrefs(Filename : STRING);	IGS	70
SaveSession(SessionFilename : STRING);	IGS	69
SelectViewport(VAR I : INTEGER) : BOOLEAN;	KPS	21
SelectWorld(I : INTEGER) : BOOLEAN;	KPS	21
SetFontAndDirection(FontUse,Direction : WORD);	BPS	36
SetSymbolSize(SymbolSizeFactor : REAL);	BPS	37
SetTextAlignment(HorizAlign : HorizAlignType; VertAlign : VertAlignType);	BPS	35
StorePointerInReal(StringPtr : POINTER; VAR X : REAL);	IGS	70
StringWidth(FontUse,Direction : WORD; TextString : STRING) : REAL;	BPS	37
StripBlanks(InString : STRING) : STRING;	BPS	33
SubHeader(HString : STRING);	BPS	37
Surface(Nx,Ny : WORD; VAR X,Y,F; TopColor,BottomColor : WORD);	3DPS	60
Transform(N : WORD; VAR InArray,OutArray; AxisName : WORD);	BPS	32
TurnClipOff;	KPS	22
TurnClipOn;	KPS	22
XErrorBar(N : WORD; VAR X1,X2,Y; T : REAL; YAxisName : WORD);	2DPS	51

Service names and parameter lists	Unit	Page no.
XSizeOfPixel : REAL;	KPS	23
YErrorBar(N : WORD; VAR X,Y1,Y2; T : REAL; YAxisName : WORD);	2DPS	51
YSizeOfPixel : REAL;	KPS	23

Appendix C - PLOTTING SHELLS

In this appendix we document two very useful utilities that provide a shell (a unit), for obtaining visual representations of arbitrary two and three dimensional functions. We provide preprogrammed templates which the user can modify with programming statements specific to the user's particular function.

The two-dimensional package is separate from the three-dimensional package, hence two independent shells are supplied. Both shells contain several common features and programming tools. Both templates contain an initialization procedure, a user function, and an optional user module. The user must write (in Turbo Pascal code) a function to plot and must specify various initialization details, such as minimum and maximum ranges and titles. A data import facility is provided to enable users to program functions dependent on a data array. For example, a likelihood function that depends on one or two parameters can easily be linked to a specific data-set.

Essentially, to produce a plot, the user must write a few lines of computer code, set some data ranges, and run the program. However, due to the flexible design of these shells, there are many technical details that users can take advantage of, such as status displays and menu commands. Furthermore, each shell exports several constants, types, and variables that may be accessed or altered by the user's program. Some of these must be altered in an initialization procedure found in the template (UInit). By reading sections C.1, C.1.1, C.2, and C.2.1, users will quickly see what can be accomplished with these two shells.

C.1. The X-Y plotting shell

By writing a function of the form

$$y = F(x)$$

and specifying a range for the variable x and the number of divisions for the range, this shell will produce an X-Y plot of the function F. Actually, the shell expands on this by allowing the user to specify a number of functions which are plotted simultaneously. The user interface or menu through which the user interacts with the program, is shown in Figure C.2 (p. 129). This screen is divided into a status section (the top) and a menu section (the bottom). The user might start by removing the function in Listing C.1 (p. 127) and replacing it with Sin(x) and altering the initialization module (UInit) appropriately.

C.1.1. The X-Y plotting shell user template

Three modules, written by the user, are provided to the plotting shell. These include a user function, a user module, and a user initilization procedure. The user function takes two parameters, **x** and **i**, where **x** is the current x data value and **i** is the current function number, and returns a REAL value. The user module is a procedure the user may write as he sees fit. It is invoked from the plot menu by selecting the U - call User module command. Finally, the user may specify data ranges, axis labels, titles, etc., in an initialization module. This module is invoked by the plotting shell at startup and also when the I - Initialize command is selected from the plot menu. Listing C.1 (p. 127) provides a sample user template implemented for a function chosen from Schnute (1985, pp. 418-419). A plot produced by this template is given in Figure C.1 (p. 128).

```
Program APL2DDem; Uses Crt,APL2D;

{$IfOpt N+} Type Real=Single; {$EndIf}

{$F+}
{*************** ADJUST THIS SECTION AS DESIRED *****************}

Function UFun(x : Real; i : Byte) : Real; {User Function}
  Var alpha,beta,gamma,S : Real;
  Begin
    alpha := 1.0; beta := 1.0; S := x;
    If (i in [2..6]) Then gamma := i-4 Else gamma := -100;
    If (gamma = 0.0) Then UFun := alpha*S*Exp(-beta*S)
    Else If ((1.0-beta*gamma*S) <= 0.0) Then UFun := 0.0
         Else UFun := alpha*S*Exp((1.0/gamma)*Ln(1.0-beta*gamma*S));
  End;

Procedure UMod; {User module} Begin End;

Procedure UInit; {User Defaults}
{*************************************************************
 *  If not specified, the defaults automatically are:        *
 *    Nx:=31; NF:=1; GraphType:=LINE;                        *
 *    x1:=0.0; x2:=1.0; yg1:=y-min; yg2:=y-max;              *
 *    TitleX:='X-Axis'; TitleY:='Y-Axis';                    *
 *    TitleG:='XY-Graph'; GDFPath := '';                     *
 *************************************************************}
 Begin
   Nx:=30; NF:=6; x1:=0.0; x2:=3.0; yg1:=0.0; yg2:=1.0;
   TitleX:='S'; TitleY:='R'; GraphType:=LINE;
   TitleG:='Recruitment'; GDFPath:=''; {e.g., C:\Pascal5}
 End;
{$F-}

Begin PlotMenu( @UFun,@UInit,@UMod ); End.
```

Listing C.1 - User template for the X-Y plotting shell

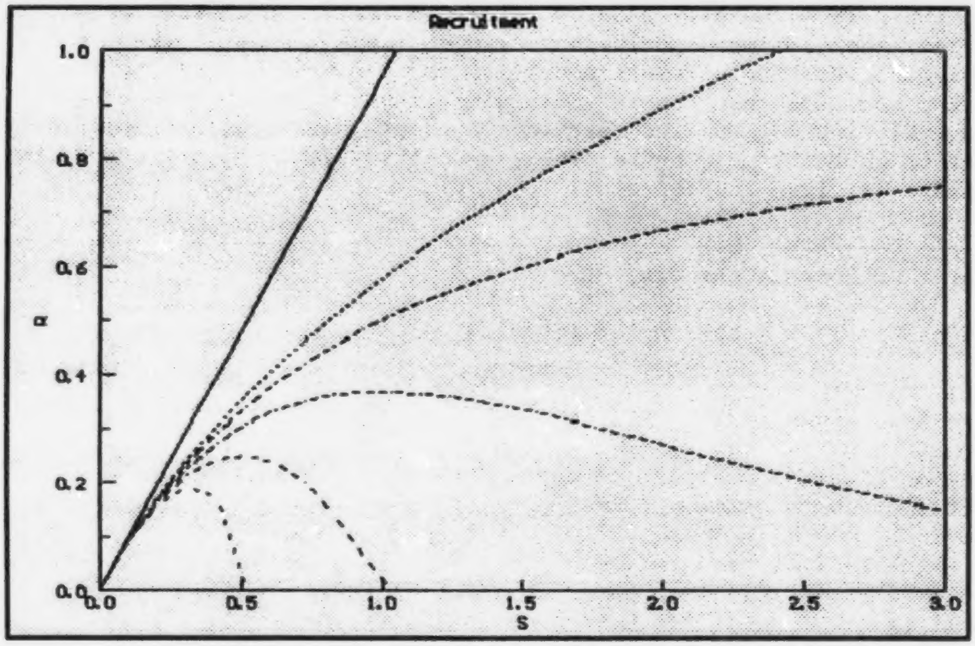
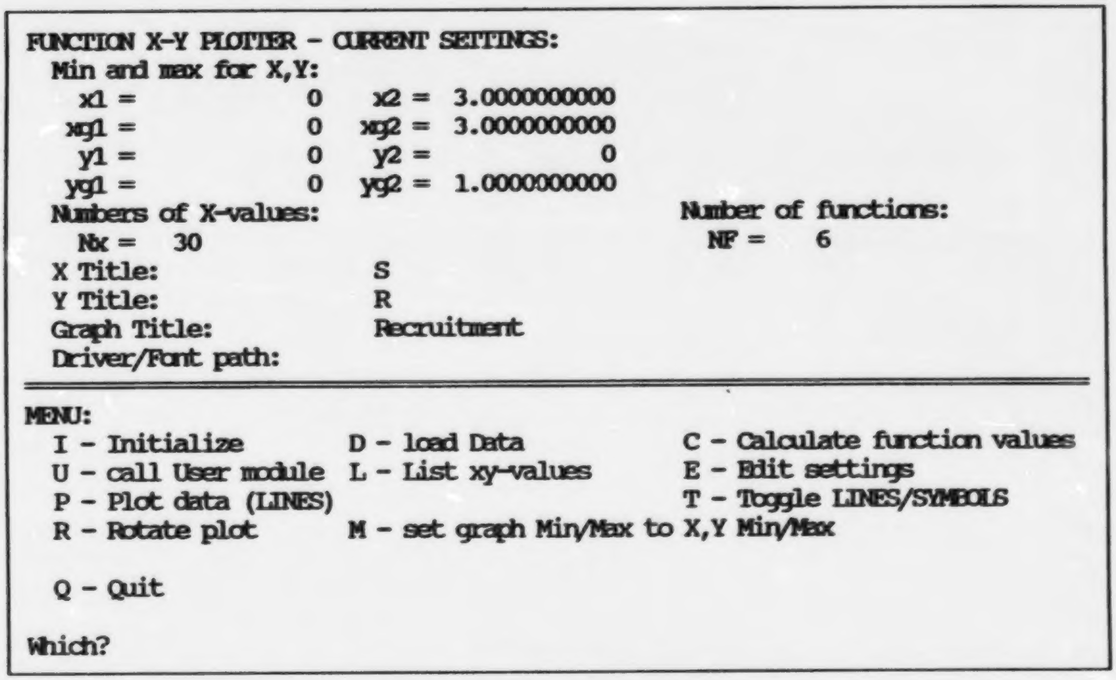


Figure C.1 - Example line graph from the X-Y plotting shell

C.1.2. The X-Y plotting shell status and menu screen

In Figure C.2 (p. 129) we see that status is maintained for the current minimum and maximum of the x and y variables (x1,x2,y1,y2) as well as the graphics minumum and maximum (xg1,xg2,yg1,yg2). The current number of points (Nx) and functions (NF) is indicated here as well. Three titles are displayed on a plot and their current value is reflected in the status section. Finally, the location of the TP graphics driver and font files (Borland 1988a) must be specified, and this is also shown.

A menu is given, listing the commands used to control various aspects of a plotting session. These commands are selected by typing the letter located in front of the "-" (dash) character in each command description.



```
FUNCTION X-Y PLOTTER - CURRENT SETTINGS:
  Min and max for X,Y:
    x1 =            0    x2 = 3.0000000000
   xg1 =            0   xg2 = 3.0000000000
    y1 =            0    y2 =            0
   yg1 =            0   yg2 = 1.0000000000
  Numbers of X-values:                          Number of functions:
    Nx =   30                                     NF =    6
  X Title:            S
  Y Title:            R
  Graph Title:        Recruitment
  Driver/Font path:
═══════════════════════════════════════════════════════════════════════
MENU:
  I - Initialize        D - load Data          C - Calculate function values
  U - call User module  L - List xy-values     E - Edit settings
  P - Plot data (LINES)                        T - Toggle LINES/SYMBOLS
  R - Rotate plot       M - set graph Min/Max to X,Y Min/Max

  Q - Quit

Which?
```

Figure C.2 - Status and Menu screen for the X-Y plotting shell

---

Here, each command is listed and briefly described:

I - Initialize: Resets all variables and calls the user initialization procedure.

D - load Data: Loads a data file into an array that the user may access. The data file must be of the form (note record five is blank):

Rec. #			
1	no. of columns		
2	"title 1"	"title 2"	"title m"
3	field width 1	field width 2	field width m
4	no. decimals 1	no. decimals 2	no. decimals m
5			
6	data 1,1	data 1,2	data 1,m
7	data 2,1	data 2,2	data 2,m
n	data n,1	data n,2	data n,m

C - Calculate function values: Calls the user function (NF * Nx) times to produce an array of function values to be used for plotting. A progress indicator is displayed on the 25th screen line.

U - call User module:	Calls the user module. The user module may perform any function at all, such as, listing a data-set loaded by the D - load Data menu command.
L - List xy-values:	Displays a list of the (x,y) points, for each function to be plotted.
E - Edit settings:	Allows the user to change all settings excluding the y1 and y2 variables, as these are the actual minimum and maximum of the user function over the x1 - x2 range.
P - Plot data:	Causes the actual plotting to occur. If the function values do not reflect the current settings, the C - Calculate function values command is invoked prior to plotting the data.
T - Toggle LINES/SYMBOLS:	Changes the (x,y) data point marking method. If LINES are currently in use, this is changed to SYMBOLS, etc.
R - Rotate plot:	This command switches the x and y axes. P - Plot data must be invoked to view this reversal.
M - set graph Min/Max to X,Y Min/Max:	Sets the xg1,xg2 and yg1,yg2 variables equal to the x1,x2 and y1,y2.
Q - Quit:	Terminates the program, returning control back to Turbo Pascal (or to DOS if run from DOS).

C.1.3. Variables and services exported by the X-Y plotting shell

There are several Turbo Pascal *const*, *type*, *var*, *function*, and *procedure* items exported by this shell. The constants set maximum ranges for (1) the number of points to plot per function, (2) the number of functions to plot, and (3) the number of rows and columns allowed in a user data-set. Three type declarations are provided to define, for the user's data-set, the internal storage format of the data matrix, variable (column) names, and display formats for each data variable. Each exported variable is accessable by the user's template program, e.g., the variable Data contains (if loaded) the user's data-set. Many of these variables should be initialized by the user's initialization procedure UInit, as indicated in brackets for each variable.

```
Const
   VarMax = 500;                     Maximum number of points to plot
   FunMax = 10;                      Maximum number of number of functions
   RowMax = 100; ColMax = 10;        Data matrix dimensions (RowMax by ColMax)

Type
  UDataVec = Array[1..RowMax,1..ColMax] of Real;      User data matrix
  VarNameVec = Array[1..ColMax] of String[10];        User var. (column) names
  FormatArr  = Array[1..2,1..ColMax] of Integer;      m:n format for each var.

Var
  Data : UDataVec;             User's data-set
  Nx,                          (UInit) Number of points to plot per function
  NF,                          (UInit) Number of functions to plot
  GDev : Integer;              Current graphics device
  x1,x2,                       (UInit) Graph data minimum/maximum range for X
  xg1,xg2,                     (UInit) Graph axis minimum/maximum range for X
  y1,y2,                       Function value minumum/maximum for Y
  yg1,yg2 : Real;              (UInit) Graph axis minimum/maximum range for Y
  TitleX,TitleY,TitleG,        (UInit) X,Y axis labels; Graph title
  GDFPath : String;            (UInit) Graphics driver and font file path
  GraphType : (LINE,SYMBOL);   (UInit) Valid graph types
  NewDat : Boolean;            Indicates whether user data is loaded
  DatName : String;            User data file name - set after data is loaded
  NVar,NRec : Integer;         Number of rows and columns in user data matrix
  VarName : VarNameVec;        User data variable or column names
  DatForm: FormatArr;          User data variable or column data formats, "m:n"

Procedure Pause;                                        Wait for <RETURN>
Function ReadKB:Char;                                   Read a keystroke
Function GetYes:Boolean;                                Read a Y or N
Procedure Tab(n:Integer);                               Write n spaces
Procedure WriteF(Var F:Text; x:Real; m,n:Integer);      Write m:n format reals
Procedure EdR(S:String; Var x:Real);                    Display S, read real x
Procedure EdI(S:String; Var x:Integer);                 Display S, read int. x
Procedure EdS(S:String; Var SS:String);                 Display S, read str. SS
Procedure WrR(S:String; x:Real);                        Display S and real x
Procedure WrI(S:String; x:Integer);                     Display S and int. x

Procedure PlotMenu( UFunPtrLoc,UInitPtrLoc,UModPtrLoc : Pointer );
```

PlotMenu is the entry point into the plotting shell. The main program section of the user's template must contain at least a call to this service. The three parameters are each pointers to the user routines **UFun**, **UInit**, and **UMod**. Listing C.1 (p. 127) demonstrates a call to this service.

C.2. The Surface/Contour plotting shell

By writing a function of the form

$$z = F(x,y)$$

and specifying a range for the x and y variables, along with the number of divisions for each range, this shell will produce both a three-dimensional surface plot and a contour plot of the function F. The user interface or menu through which the user interacts with the program, is shown in Figure C.5 (p. 136). This screen is divided into a status section (the top) and a menu section (the bottom).

C.2.1. The user template for the Surface/Contour plotting shell

Three modules are provided to the plotting shell by the user. These include a user function, a user module, and a user initilization procedure. The user function takes two parameters, **x** and **y**, where **x** is the current x data value and **y** is the current y data value, and returns a REAL value. The user module is a procedure the user may use as he wishes. It is invoked from the plot menu by selecting the U - call User module command. Listing C.2 (p. 133) uses the user module to list the contents of the user's data-set. Finally, the user may specify data ranges, axis labels, titles, etc., in an initialization module. This module is invoked by the plotting shell at startup and also when the I - Initialize command is selected from the plot menu.

Listing C.2 (p. 133) provides a sample user template preprogrammed to plot the log-likelihood for the Von Bertalanffy growth model (1) (Mittertreiner and Schnute 1985, p. 1)

$$(1) \quad y(t) = y_\infty[1 - e^{-K(t-t_0)}] \quad ,$$

where y(t) is the length at age t and (y_∞,K,t_0) is a vector of three parameters. Using data (APL3DDem.DAT) selected from Mittertreiner and Schnute (1985, pp. 1-2) we know the minimum is found when the parameter vector equals (57.3,0.16,0.15). For our example we set t_0 to 0 and vary y_∞ from 50.0 to 70.0 and k from 0.05 to 0.3. Notice that our choice of t_0 is "close" to the actual minimum. Incidentally, we set t_0 to 0 because we can't plot four-dimensional functions. Using the data-set we plot a surface diagram and contour plot using the function

$$(2) \quad F(y_\infty,K) = n \cdot \mathrm{Log}\Big(\sum_{t=1}^{n} [y_\infty(1 - e^{-K(t-t_0)}) - y(t)]^2 \Big) \quad ,$$

where n is the number of data points (here 16). To produce the surface and contour plots are shown in Figures C.3 (p. 134) and C.4 (p. 135), the user must first load the data (APL3DDem.DAT), then calculate the function values, and finally, execute the contour or surface plot options. These actions are performed from the menu shown in Figure C.5 (p. 136).

```
Program APL3DDem; Uses Crt,APL3D;

{$IfOpt N+} Type Real=Single; {$EndIf}

{$F+}
{**************** ADJUST THIS SECTION AS DESIRED *****************}

Function UFun(x,y : Real) : Real; {User Function}
  Var L_inf,K,t_zero,D,SS : Real; i : Integer;
  Begin
    L_inf := x; K := y; t_zero := 0.0; SS := 0.0;
    For i := 1 to NRec do
      Begin
        D := (L_inf*(1.0-Exp(-K*(Data[i,1]-t_zero))) - Data[i,2]);
        SS := SS+D*D;
      End;
    UFun := NRec*Ln(SS);
  End;

Procedure UMod; {User module} Var i : Integer; {Display loaded data}
  Begin
    ClrScr; WriteLn('Current data-set:');
    WriteLn('Rec# ',VarName[1]:12,VarName[2]:12);
    For i := 1 to NRec do WriteLn(i:3,': ',Data[i,1]:12:3,Data[i,2]:12:3);
    Pause;
  End;

Procedure UInit; {User Defaults}
{***************************************************************
 *  If not specified, the defaults automatically are:          *
 *    Nx:=31; Ny:=31; Nc:=7; RelAbsZ := RELZ;                  *
 *    x1:=0.0; x2:=1.0; y1:=0.0; y2:=1.0; zg1:=0.0; zg2:=0.0;  *
 *    TitleX:='X-Axis'; TitleY:='Y-Axis'; TitleZ:='Z-Axis';    *
 *    TitleG:='XYZ-Graph'; Reverse:=False; GDFPath := '';      *
 ***************************************************************}
  Begin
    Nx:=31; Ny:=31; Nc:=7; RelAbsZ := RELZ;
    x1:=50.0; x2:=70.0; y1:=0.05; y2:=0.3; zg1:=0.0; zg2:=1.0;
    TitleX:='L-infinity'; TitleY:='K'; TitleZ:='Likelihood';
    TitleG:='Von Bertalanffy growth model'; GDFPath := '';
  End;
{$F-}

Begin SurfContMenu( @UFun,@UInit,@UMod ); End.
```

Listing C.2 - User template for the Surface/Contour plotting shell

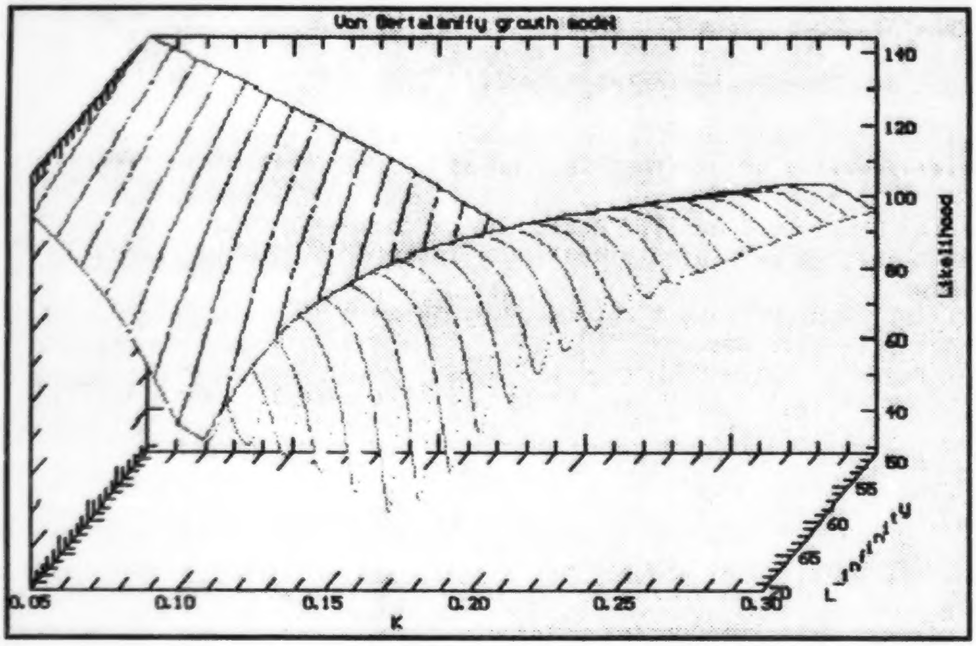


Figure C.3 - Example surface diagram from the Surface/Contour plotting shell

Notice that the lowest point of the above surface occurs when y_∞ (L-infinity) equals 57.3 and K equals 0.16. This figure, when viewed on a color computer display screen, distinguishes between the top and bottom parts of the surface by using two different colors: CYAN for the top and BLUE for the bottom.

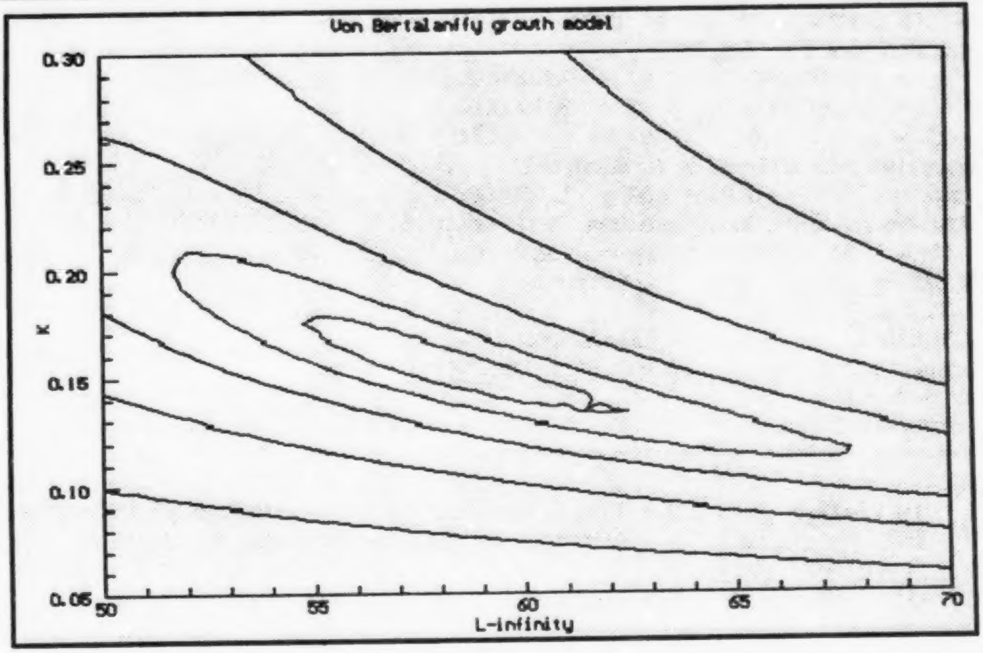


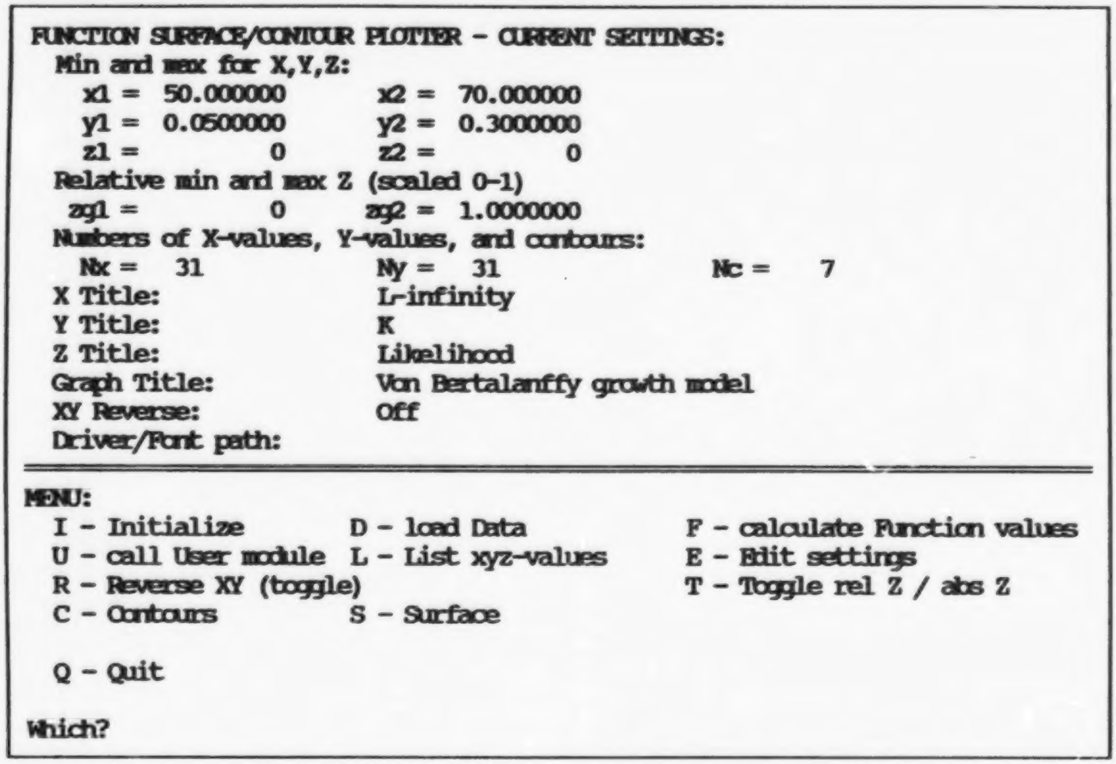
Figure C.4 - Example contour plot from the Surface/Contour plotting shell

Notice that the interior contour appears to be composed of two parts. This effect is due to the particular grid size used in this example. This problem may be overcome by increasing the grid size and therefore the contouring resolution.

C.2.2. The Surface/Contour plotting shell status and menu screen

In Figure C.5 (p. 136) we see that status is maintained for the current minimum and maximum of the x, y, and z variables (x1,x2,y1,y2,z1,z2) as well as the graphics minimum and maximum for z (zg1,zg2). The zg1,zg2 values reflect either a relative scale (0-1) or an absolute scale for z. The current number of points (Nx,Ny) and contour lines (Nc) are indicated here as well. Four titles are displayed on a plot and their current value is shown in the status section. The current X-Y orientation is reflected by an **Off** or **On** beside the string **XY Reverse**. Finally, the location of the TP graphics driver and font files (Borland 1988a) must be specified and this is also shown.

A menu is given, listing the commands used to control various aspects of a plotting session. These commands are selected by typing the letter located in front of the "-" (dash) character in each command description.



```
FUNCTION SURFACE/CONTOUR PLOTTER - CURRENT SETTINGS:
  Min and max for X,Y,Z:
    x1 =  50.000000       x2 =  70.000000
    y1 =  0.0500000       y2 =  0.3000000
    z1 =          0       z2 =          0
  Relative min and max Z (scaled 0-1)
   zg1 =          0      zg2 =  1.0000000
  Numbers of X-values, Y-values, and contours:
    Nx =   31             Ny =   31              Nc =    7
  X Title:                L-infinity
  Y Title:                K
  Z Title:                Likelihood
  Graph Title:            Von Bertalanffy growth model
  XY Reverse:             Off
  Driver/Font path:
═══════════════════════════════════════════════════════════════════
MENU:
  I - Initialize        D - load Data           F - calculate Function values
  U - call User module  L - List xyz-values     E - Edit settings
  R - Reverse XY (toggle)                       T - Toggle rel Z / abs Z
  C - Contours          S - Surface

  Q - Quit

Which?
```

Figure C.5 - Status and Menu screen for the Surface/Contour plotting shell

---

Here, each command is listed and briefly described:

I - Initialize: Resets all variables and calls the user initialization procedure.

D - load Data: * See section C.1 *

F - calculate Function values: Calls the user function (Nx * Ny) times to produce an array of function values to be used for plotting. Also, the (Nc) contour lines are calculated. A progress indicator is displayed on the 25th screen line.

U - call User module: * See section C.1 *

L - List xyz-values: Displays a list of the (x,y,z) points.

E - Edit settings: Allows the user to change all settings excluding the **z1** and **z2** variables, as these are the actual **minimum** and **maximum** of the user function over the **x1** - **x2** and **y1** - **y2** range.

R - Reverse XY (toggle): This command switches the x and y axes.

T - Toggle rel Z / abs Z: Toggles the interpretation of **zg1** and **zg2** between RELATIVE and ABSOLUTE.

C - Contours: Causes a contour plot to be displayed. If the function values do not reflect the current settings, the **F - calculate Function values command** is invoked prior to plotting the contours.

S - Surface: Causes a three-dimensional surface diagram to be displayed. If the function values do not reflect the current settings, the **F** - **calculate Function values command** is invoked prior to drawing the surface.

Q - Quit: Terminates the program, returning control back to Turbo Pascal (or to DOS if run from DOS).

C.2.3. Variables and services exported by the Surface/Contour plotting shell

There are several Turbo Pascal *const*, *type*, *var*, *function*, and *procedure* items exported by this shell. See section C.1.3. for a more indepth discussion of this topic.

```
Const
  VarMax = 41;                      Maximum number of X and Y points
  VarMaxSq=1681;                    Maximum number of X-Y Grid values
  RowMax = 100; ColMax = 10;        Data matrix dimensions (RowMax by ColMax)

Type
  UDataVec = Array[1..RowMax,1..ColMax] of Real;      User data matrix
  VarNameVec = Array[1..ColMax] of String[10];        User var. (column) names
  FormatArr  = Array[1..2,1..ColMax] of Integer;      m:n format for each var.
```

```
Var
  Data : UDataVec;         User's data-set
  Nx,Ny,                   (UInit) Number of X and Y points
  Nc,                      (UInit) Number of contour lines
  GDev : Integer;          Current graphics device
  x1,x2,                   (UInit) Graph data minimum/maximum range for X
  y1,y2,                   (UInit) Graph data minimum/maximum range for Y
  z1,z2 : Real;            Function minimum/maximum for Z
  zg1,zg2 : Real;          (UInit) Relative or absolute Graph axis min/max for Z
  TitleX,TitleY,TitleZ,    (UInit) X,Y,Z axis labels
  TitleG,                  (UInit) Graph title
  GDFPath : String;        (UInit) Graphics driver and font file path
  Reverse : Boolean;       (UInit) If true, reverse X and Y axes
  RelAbsZ : (RELZ,ABSZ);         (UInit) Indicates interpretation of zg1,zg2
  NewDat : Boolean;              Indicates whether user data is loaded
  DatName : String;              User data file name - set after data is loaded
  NVar,NRec : Integer;           Number of rows and columns in user data matrix
  VarName : VarNameVec;          User data variable or column names
  DatForm: FormatArr;            User data variable or column data formats, "m:n"

Procedure Pause;                                         Wait for <RETURN>
Function ReadKB:Char;                                    Read a keystroke
Function GetYes:Boolean;                                 Read a Y or N
Procedure Tab(n:Integer);                                Write n spaces
Procedure WriteF(Var F:Text; x:Real; m,n:Integer);       Write m:n format reals
Procedure EdR(S:String; Var x:Real);                     Display S, read real x
Procedure EdI(S:String; Var x:Integer);                  Display S, read int. x
Procedure EdS(S:String; Var SS:String);                  Display S, read str. SS
Procedure WrR(S:String; x:Real);                         Display S and real x
Procedure WrI(S:String; x:Integer);                      Display S and int. x

Procedure SurfContMenu( UFunPtrLoc,UInitPtrLoc,UModPtrLoc : Pointer );
```

SurfContMenu is the entry point into the plotting shell. The main program section of the user's template must contain at least a call to this service. The three parameters are each pointers to the user routines UFun, UInit, and UMod. Listing C.2 (p. 133) demonstrates a call to this service.

Appendix D - ERROR MESSAGES

This appendix lists error messages generated by all CGP, IGS, CPP, and MRPGS functions and procedures. This does not include TP or DOS error messages.

CGP Message	Unit	Function or Procedure
Attempt to take 10 to a power > 38.17 -- returns missing data value	CGPHLU	TenE
Attempt to take Log10 of a number < 0 or > missing data value -- returns 0.0	CGPHLU	Log10
Bot > Top in call to Axis - exchanged them	CGPHLU	Axis
Extreme out of range for Log axis	CGPHLU	Axis
Extreme out of range for Prb axis.	CGPHLU	Axis
All missing values in call to PlotLine	CGPHLU2D	PlotLine
TotBars = 0 in call to PlotBar	CGPHLU2D	PlotBar
All missing values in call to PlotBar	CGPHLU2D	PlotBar
NBin = 0 in call to PlotHistoLine	CGPHLU2D	PlotHistoLine
All missing values in call to PlotHistoLine	CGPHLU2D	PlotHistoLine
Unsuccessful LoadDriver with FileString = 'filename.ext'	CGPLLU	LoadDriver
Error registering driver:'other message'	CGPLLU	LoadDriver
Unsuccessful LoadFont with FileString = 'filename.ext'	CGPLLU	LoadFont
Error registering font:'other message'	CGPLLU	LoadFont
Nx < 1 in call to DefineRegion - set equal to 1	CGPLLU	DefineRegion
Nx > 2 in call to DefineRegion - set equal to 2	CGPLLU	DefineRegion
Ny < 1 in call to DefineRegion - set equal to 1	CGPLLU	DefineRegion
Ny > 2 in call to DefineRegion - set equal to 2	CGPLLU	DefineRegion
Rx1 >= Rx2 OR Ry1 >= Ry2 in call to DefineViewport - no viewport defined	CGPLLU	DefineViewport
Rx1 or Rx2 or Ry1 or Ry2 out of 0.0 - 1.0 range - no viewport defined	CGPLLU	DefineViewport
Rx1 >= Rx2 OR Ry1 >= Ry2 in call to DefineWorld no world defined	CGPLLU	DefineWorld
Cannot detect a graphics card	CGPLLU	InitLowLevel
Cannot find driver file	CGPLLU	InitLowLevel
Invalid driver	CGPLLU	InitLowLevel
Insufficient memory to load driver	CGPLLU	InitLowLevel
Invalid graphics mode for selected driver	CGPLLU	InitLowLevel
Invalid device number	CGPLLU	InitLowLevel

IGS Message		
*** ERROR: Can't open session files	IGSMain	SaveSession or LoadSession
*** ERROR: Cannot open specified data file	IGSMain	SaveSession or LoadSession
*** ERROR: Cannot read data file	IGSMain	SaveSession or LoadSession
*** ERROR: Cannot access data file	IGSMain	SaveSession or LoadSession
*** ERROR: Too many data fields	IGSMain	SaveSession or LoadSession
*** ERROR: Not enough data fields	IGSMain	SaveSession or LoadSession
*** ERROR: Open quote data field	IGSMain	SaveSession or LoadSession
*** ERROR: Invalid data field	IGSMain	SaveSession or LoadSession
*** ERROR: Cannot allocate memory	IGSMain	SaveSession or LoadSession
*** ERROR: Empty field	IGSMain	SaveSession or LoadSession
*** ERROR: Wrong field type	IGSMain	SaveSession or LoadSession
Selected range exceeds worksheet size -- Clip (Y/N)?	IGSMain	SelectRange
Range name not defined: 'name'	IGSIntrp	FindRangeName
Named range 'name' not consistent with spreadsheet size	IGSIntrp	FindRangeName
Attempt to treat real value as a label. Range name: 'name'	IGSIntrp	LoadArray: GetCell

IGS Message (continued...)

Attempt to extract numeric value from a label column. Range name: 'name'	IGSIntrp	LoadArray: GetCell
Number of heap allocations exceeds limit	IGSIntrp	TestGetMem
Insufficient memory for plot variable allocation	IGSIntrp	TestGetMem
No data ranges supplied	IGSIntrp	SizeAndAllocate
Unequal number of X and Y point - lesser used	IGSIntrp	DoXYPlot
Number of point labels <> number of points — lesser used	IGSIntrp	DoXYPlot
No data value ranges specified	IGSIntrp	DoBarChart
Numbers of elements in data ranges do not equal number of primary categories	IGSIntrp	DoBarChart
No extrema specified for histogram plot	IGSIntrp	DoHistogram
Unequal number of X and Y points - lesser used	IGSIntrp	DoHistogram
Number of data points does not equal number of intervals — lesser used	IGSIntrp	DoHistogram
Unequal number of X,Y,Z points — least used	IGSIntrp	DoLollipop
No symbol specified	IGSIntrp	DoLollipop
Number of point labels <> number of points — lesser used	IGSIntrp	DoLollipop
Image buffer allocation failure — do not use screen regions	IGSIntrp	GetImageBuffers
Attempt to use unallocated image buffer(s)	IGSIntrp	BufferImage
No graphics hardware detected	IGSIntrp	DrawGraph
Invalid graph mode	IGSIntrp	DrawGraph

CPP Message

*** ERROR: Can't open 'filename.ext'	PLT	NewDataFile
*** ERROR: Too many data fields in row 'cellrow'	PLT	NewDataFile
*** ERROR: Not enough data fields in row 'cellrow'	PLT	NewDataFile
*** ERROR: Open quote in row 'cellrow', col 'cellcol'	PLT	NewDataFile
*** ERROR: Empty field in row 'cellrow', col 'cellcol'	PLT	NewDataFile
*** ERROR: Invalid data field in row 'cellrow', col 'cellcol'	PLT	NewDataFile

MRPGS Message

Unable to load definitions	MRPlot	main program
Unable to interpret recovery field 'fieldnum' as an integer	MRProc	GetRealData: RealRecField
Unable to interpret release field 'fieldnum' as an integer	MRProc	GetRealData: RealRelField
No sort fields specified	MRProc	MRProcess
Recovery sort field specified with release data field	MRProc	MRProcess
Unable to open release file	MRProc	MRProcess
Unable to open recovery file	MRProc	MRProcess
Problem too large	MRProc	MRProcess
Required spreadsheet array size exceeds maximum array size allowed	MRProc	MRProcess
Not enough space on heap for spreadsheet array	MRProc	MRProcess
No Tag code in release file corresponding to the recovery file Tag code 'code'	MRProc	MRProcess

Appendix E - DISTRIBUTION DISKS

The complete CGP (includes the IGS, CPP, MRPGS, and demo programs) and the PRT is deliverable on one 1.2 MByte floppy or four 360 KByte floppies. Most of the files are compressed into a space saving format and must be decompressed before use. These files, known as "self extracting archive files" (PKWare 1988), are easily identified by the "!" character appended to the disk file name and the suffix ".EXE".

A "read me" file named Read.ME resides on the first diskette. This file should be read first for a list of any last minute changes, the current file list, and instructions for copying and installing the package. Given below is a list of each disk file with a brief description indicating its purpose. A list of all the files supplied in the package is also given.

Filename	Description
APL!.EXE	Example applications of 2DPS and 3DPS services.
CGP!.EXE	All CGP and IGS Pascal source code files.
Demo2D!.EXE	2D demonstration source code and executable files.
Demo3D!.EXE	3D demonstration source code and executable files.
MRP!.EXE	All MRPGS Pascal source code and executable files.
PLT!.EXE	All CPP Pascal source code and executable files.
PRT!.EXE	All PrintImage Pascal source code and executable files.
Read.ME	Important instructions.
CGA.BGI	CGA graphics card driver file.
EGAVGA.BGI	EGA and VGA graphics cards driver file.
HERC.BGI	HGC graphics card driver file.
GOTH.CHR	Gothic stroked-font file.
LITT.CHR	Small stroked-font file.
SANS.CHR	Sanserif stroked-font file.
TRIP.CHR	Triplex stroked-font file.

The complete package requires approximately 2.5 MBytes of disk space. Listed below are all the files grouped into categories (excluding the BGI and CHR files listed above).

Filename	Description or Comment
APL3DDEM.PAS	Surface and Contour program. User writes a function and the program displays its surface view and contour lines (a sample function is provided).
APL3D.PAS	Utilities and menus provided for APL3DDEM.PAS.
APL2DDEM.PAS	Multiple line/symbol plotting program. User writes function(s) and the program displays the graph.
APL2D.PAS	Utilities and menus provided for APL2DDEM.PAS.
CGPBPS.PAS	CGP Basic Plotting Services.
CGPHCS.PAS	CGP Hard Copy Services.
CGP2DPS.PAS	CGP 2-dimensional Plotting Services.
CGP3DPS.PAS	CGP 3-dimensional Plotting Services.
CGPKPS.PAS	CGP Kernel Plotting Services.
DEMO*.PAS,.EXE	Files prefixed by "DEMO" give short demonstrations of the CGP capabilities.
IGSMain.PAS	IGS Main Menu Unit. This contains the entry point into the IGS.
IGS*.PAS	Files prefixed by "IGS" are IGS support units.
PLT.PAS,.EXE	CPP program file.
PLTFile.PAS	CPP support unit.
MRPlot.PAS,.EXE	MRPGS program file.
MRP*.PAS	Files prefixed by "MRP" are MRPGS support unit files.
PRTCGA.PAS,.EXE	PrintImage support for the CGA graphics card.
PRTEGA.PAS,.EXE	PrintImage support for the EGA graphics card.
PRTHerc.PAS,.EXE	PrintImage support for the HGC graphics card.
PRT.PAS	The procedure (unit) version of PrintImage.

References

Abramowitz and Stegun. 1965. Handbook of Mathematical Functions. U.S. Department of Commerce, National Bureau of Standards, Applied Mathematics Series. 55: 933.

Borland. 1988a. Turbo Pascal 5.0 User's Guide. Borland International Inc. 10 9 8 7 6 5 4 3 2 1: xiii + 350 p.

Borland. 1988b. Turbo Pascal 5.0 Reference Guide. Borland International Inc. 10 9 8 7 6 5 4 3 2 1: xvii + 493 p.

Bourke, P.D. 1987. A Contouring Subroutine. Byte Magazine. 12(6): 143-150.

Kuhn, B.R., L. Lapi, and J.M. Hamer. 1988. An Introduction to the Canadian database on marked Pacific salmonids. Can. Tech. Rep. Fish. Aquat. Sci. 1649: viii + 56 p.

Kuhn, B.R. 1988. The MRP-Reporter program: A data extraction and reporting tool for the Mark Recovery Program database. Can. Tech. Rep. Fish. Aquat. Sci. 1625: 145 p.

Mittertreiner, A.C. and J. Schnute. 1985. SIMPLEX: A manual and software package for easy nonlinear parameter estimation and interpretation in fishery research. Can. Tech. Rep. Fish. Aquat. Sci. 1384: 90 p.

PKWare. 1988. PKWare Data compression utilities. PKWare Inc., 7032 Ardara Avenue, Glendale, WI. 53209. iii + 31 p.

Schnute, J. 1985. A general theory for analysis of catch and effort data. Can. J. Fish. Aquat. Sci. 42: 414-429.

TurboPower. 1987. Turbo Professional 4.0 User's Manual. TurboPower Software. Chapter 5: 1-47.

Acknowledgements

We would like to thank Dr. J. Schnute, Dr. D. Welsh, and S. McKinnell for their valuable comments and suggestions. Also, special thanks to Dan Nesbitt who played a very significant role in the development of the graphics image printing software.

Also, Dr. Schnute developed the initial prototype of the Surface/Contour plotting shell. From this prototype, we created the X-Y plotting shell and the current Surface/Contour plotting shell. We emphasize here, our appreciation for Dr. Schnute's significant contributions to this package.

Index

←↑↓→ 74, 116
<BackSpace> 75
<Ctrl-Left-Arrow> 75
<Ctrl-PgDn> 77
<Ctrl-PgUp> 77
<Ctrl-Right-Arrow> 75
<Delete> 75
<End> 74
<Enter> 74
<Esc> 74
<F1> 75
<Home> 74
<Insert> 75
3D scatter plots 92
80287 coprocessor 9
    emulation 9
    Turbo Pascal version 5.0 9

- A -

AccumulateHistogram 48
Aspect ratio
    stroked-font 12
    symbols 12
Axis 30
    continuous variables 30
    data transformations 32
    discrete variables (AxisD) 31
    logarithmic 31
    minor tick marks 31
    probability 31
    setup 12
AxisD 31

- B -

bit-mapped font 10

- C -

CharDepth 36
CharHeight 36
Circle 26
clipping 11, 22
ColFmtRec 68
color control 13
    BoxColor 29
    FieldColor 29
    LabelColor 29
    VPBoxColor 29, 44
ColTitle 68
Constants
    MaxCols 67
    MaxColTitleLen 67
    MaxContourLineNdx 43
    MaxErrorText 16
    MaxLegendNum 42
    MaxNumViewports 16
    MaxNumWorlds 16
    MaxPCats 42
    MaxPolySize 16
    MaxSCats 42
    MinBarWidthInPixels 42
continuous variables 12, 42
    PlotBar 45
    PlotLine 44
    PlotSymbol 45
Contour 52
CornerBox 59
CPP
    column title record 100
    Continue 98
    data files 99
    data record 99
    Different data 97
    display format record 99
    Exit 98
    File Menu 96
    IGS 97
    Load session 98
    New session 96
    Plot.X00 97
    Plot.X01 97
    Plot.X02 97
    Preferences 98
    PREFile.WKS 98

- D -

DataFieldType 67
DefineRegion 20
DefineViewport 20
DefineWorld 21
discrete variables 12, 42

  - PlotBarD 47
  - PlotBarS 47
  - PlotHistoBar 50
  - PlotHistoLine 49
  - PlotHistoSymbol 50
  - PlotSymbolD 46
- Discrete-Continuous plot 89
- DrawVPBox 21

- E -

- error bars 51
- error handling 10
  - ErrorText 16
  - NErrors 16
  - PrintError 17
  - RecordError 17
  - ResetError 17

- F -

- File Menu 96
- FontPath 34
- Fonts 34
  - bit-mapped 10
  - Default 82
  - directory path 34
  - font name constants 34
  - font use name constants 34
  - LoadFont 19
  - mapping 33
  - stroked-font 10, 12, 19
  - stroked-fonts 26
  - use and direction 36

- G -

- Graph Menu 80
  - Draw 80
  - Edit 80
  - New 80
- graph mode 83
- graphics driver 17
- graphics mode 18

- H -

- hard copy printing 15, 17
- HardCopyOff 19
- HardCopyOn 19
- Histogram 49
- Histogram plot 90

- I -

- IGS
  - ←↑↓→ 74
  - <BackSpace> 75
  - <Ctrl-Left-Arrow> 75
  - <Ctrl-PgDn> 77
  - <Ctrl-PgUp> 77
  - <Ctrl-Right-Arrow> 75
  - <Delete> 75
  - <Dn-Arrow> 85
  - <End> 74
  - <Enter> 74
  - <Esc> 74
  - <F1> 75, 84
  - <Home> 74
  - <Insert> 75
  - <Up-Arrow> 85
  - 3D scatter plots 92
  - ColFmtRec 68
  - ColTitle 68
  - DataFieldType 67
  - Discrete-Continuous plot 89
  - error message color 78
  - Exit 79
  - fonts 82
  - Graph Menu 80
  - graph mode 83
  - heading color 78
  - highlight color 78
  - Histogram plot 90
  - Lollipop plot 92
  - numeric field 74
  - preferences 70
  - range field 74, 75
  - ranges 76
  - releasing memory 69
  - rotary field 74, 75
  - SavePrefs 79
  - SaveSession 79
  - text color 78
  - text field 74
  - Wks border color 78
  - X-Y plot 85
- IGSMainMenu 66
- IGSMissingData 65

IGSRelease 69
image printing 15
InitCGP 29
initialization 9, 11, 17, 29, 33
  graphics driver 17
  graphics mode 18
  InitCGP 29
  InitKPS 17
InitKPS 17
installing 5

- L -

label points 51
LabelLolliPoints 60
LabelPoints 51
LabelString 29
lines 23, 24, 45
  pattern 24
  thickness 23, 24
Load session 98
LoadDriver 19
LoadFont 19
LoadPrefs 70
LoadSession 70
Log10 32
Lolliplot 59
Lollipop plot 92

- M -

MainHeader 37
MapFont 33
MATH chip 9, 29
  Turbo Pascal version 5.0 9
MaxCols 67
MaxColTitleLen 67
MaxContourLineNdx 43
MaxErrorText 16
MaxInArray 32
MaxLegendNum 42
MaxNumViewports 16
MaxNumWorlds 16
MaxPCats 42
MaxPolySize 16
MaxSCats 42
memory requirements 121
MinBarWidthInPixels 42
minimum system configuration 121
MinInArray 32
missing data 10
MissingTestValue 29
MRP database 102
  IGS 102
  MRP-Reporter 102
MRPGS 103
  <ESC> 105
  data types 105
  Load definitions 103
  Main Menu 103
  MRP flat files 104
  sort fields 107
  sort levels 104
  spreadsheet 105, 106

- N -

NArc 25
NArcCoordsType 16
NBar 24
NBar3D 24
NCircle 26
NClearViewport 21
NDrawPoly 25
NEllipse 26
New session 96
NextRegion 20
NFillPoly 25
NFloodFill 25
NGetArcCoords 25
NGetPixel 22
NGetX 22
NGetY 22
NLine 23
NLineRel 23
NLineTo 23
NMoveRel 22
NMoveTo 22
Normalized Device Coordinates 11
NOutText 26
NOutTextXY 26
NPieSlice 26
NPutPixel 22
NRectangle 24
NSetLineStyle 24
NSetUserCharSize 26
NTextHeight 27
NTextWidth 27

- numeric field 74

- O -

- OutAText 35
- OutATextXY 35
- OuterFrame 38

- P -

- PKWare 141
- PlaceSymbol 38
- PlaneBox 43
- plot layout 43
  - 3D plots 59
- Plot.X00 97
- Plot.X01 97
- Plot.X02 97
- PlotBar 45
- PlotBarD 47
- PlotBarS 47
- PlotHistoBar 50
- PlotHistoLine 49
- PlotHistoSymbol 50
- PlotLine 44
- PlotSymbol 45
- PlotSymbolD 46
- point label
  - 3D scatter plots 60
- preferences 70, 78
- PREFile.WKS 98
- PrintError 17
- PrintImage 114
  - ←↓↑→ 116
  - <Ctrl><RightShift><PrtScr> 113
  - <Enter> 116
  - <ESC> 116
  - graphics cards and modes 111
  - hard-wired defaults 116
  - help screen 113
  - Hercules Graphics Card 116
  - installing/calling 113
  - last-run defaults 116
  - memory requirements 110
  - Menu 116
  - printer mode 118
  - printer port 118
  - PRT.TPU 110
  - removing from memory 119
  - scaling factors 119
  - scan type (direction) 118
  - unit version 114
- Procedures and Functions
  - AccumulateHistogram 48
  - Axis 30
  - AxisD 31
  - CharDepth 36
  - CharHeight 36
  - Contour 52
  - CornerBox 59
  - DefineRegion 20
  - DefineViewport 20
  - DefineWorld 21
  - DrawVPBox 21
  - FontPath 34
  - HardCopyOff 19
  - HardCopyOn 19
  - IGSMainMenu 66
  - IGSRelease 69
  - InitCGP 29
  - InitKPS 17
  - LabelLolliPoints 60
  - LabelPoints 51
  - LoadDriver 19
  - LoadFont 19
  - LoadPrefs 70
  - LoadSession 70
  - Log10 32
  - Lolliplot 59
  - MainHeader 37
  - MapFont 33
  - MaxInArray 32
  - MinInArray 32
  - NArc 25
  - NBar 24
  - NBar3D 24
  - NCircle 26
  - NClearViewport 21
  - NDrawPoly 25
  - NEllipse 26
  - NextRegion 20
  - NFillPoly 25
  - NFloodFill 25
  - NGetArcCoords 25
  - NGetPixel 22
  - NGetX 22
  - NGetY 22

NLine 23
NLineRel 23
NLineTo 23
NMoveRel 22
NMoveTo 22
NOutText 26
NOutTextXY 26
NPieSlice 26
NPutPixel 22
NRectangle 24
NSetLineStyle 24
NSetUserCharSize 26
NTextHeight 27
NTextWidth 27
OutAText 35
OutATextXY 35
OuterFrame 38
PlaceSymbol 38
PlaneBox 43
PlotBar 45
PlotBarD 47
PlotBarS 47
PlotHistoBar 50
PlotHistoLine 49
PlotHistoSymbol 50
PlotLine 44
PlotSymbol 45
PlotSymbolD 46
PrintError 17
PrintImage 114
RecordError 17
ResetError 17
SavePrefs 70
SaveSession 69
SelectViewport 21
SelectWorld 21
SetFontAndDirection 36
SetSymbolSize 37
SetTextAlignment 35
StorePointerInReal 70
StringWidth 37
StripBlanks 33
SubHeader 37
Surface 60
Transform 32
TurnClipOff 22
TurnClipOn 22
XErrorBar 51
XSizeOfPixel 23
YErrorBar 51
    YSizeOfPixel 23
PRTCGA.EXE 110
PRTEGA.EXE 110
PRTHerc.EXE 110

- R -

Range
    . 77
    <BackSpace> 77
    <Ctrl-PgDn> 77
    <Ctrl-PgUp> 77
    <Dn-Arrow> 77
    <End> 77
    <Enter> 77
    <Esc> 77
    <Home> 77
    <Left-Arrow> 77
    <Right-Arrow> 77
    <Up-Arrow> 77
range field 74, 75
ranges 76
    list 77
    select 76
Read.ME 141
RecordError 17
ResetError 17
rotary field 74, 75

- S -

SavePrefs 70
SaveSession 69
screen regions 11
    DefineRegion 20
    NextRegion 20
SelectViewport 21
SelectWorld 21
SetFontAndDirection 36
SetSymbolSize 37
SetTextAlignment 35
spreadsheet 62, 65, 76, 81
stacked bar charts 47
Stacked bars 48
StorePointerInReal 70
string pointers 70
StringWidth 37
StripBlanks 33
stroked-font 10, 12, 19
    LoadFont 19

stroked-fonts 26
SubHeader 37
Surface 60
symbols 37, 45, 47
    size 37

- T -

text field 74
text writing 26, 33, 35
    alignment point 35
    center on point 35
    character size 26
    get text size 27
    left justified 35
    right justified 35
    text size 36
Transform 32
transformations 32
    logarithmic 42
    probability 42
Turbo Pascal 5, 15
    compiler settings 121
TurnClipOff 22
TurnClipOn 22

- V -

viewport 11, 20, 21, 38, 44
    DefineViewport 20
    DrawVPBox 21
    NClearViewport 21
    SelectViewport 21

- W -

world coordinate system 11, 21
    DefineWorld 21
    pixel size 23
    SelectWorld 21
    viewports 11

- X -

X-Y plot 85
X-Y-Z scatter plots 60
XErrorBar 51
XSizeOfPixel 23

- Y -

YErrorBar 51
YSizeOfPixel 23